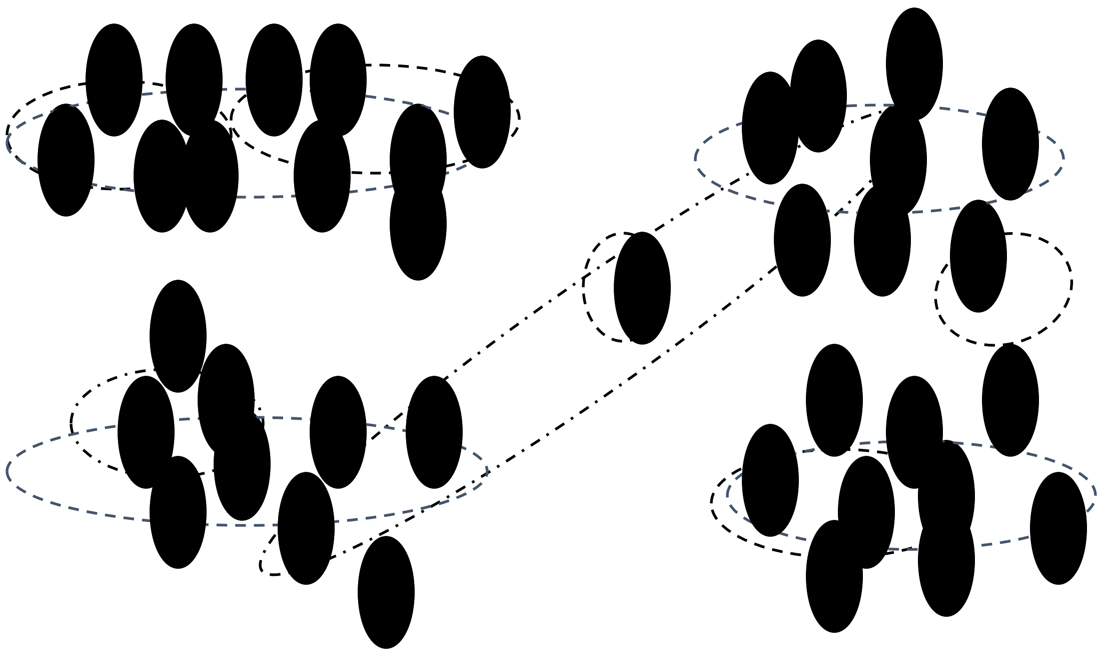
Clustering Examples

- ***Segment*** customer database based on similar buying patterns.
- Group houses in a town into neighborhoods based on similar features.
- Identify new plant species
- Identify similar Web usage patterns

Clustering Example

Income	Age	Children	Marital Status	Education
$25,000	35	3	Single	High School
$15,000	25	1	Married	High School
$20,000	40	0	Single	High School
$30,000	20	0	Divorced	High School
$20,000	25	3	Divorced	College
$70,000	60	0	Married	College
$90,000	30	0	Married	Graduate School
$200,000	45	5	Married	Graduate School
$100,000	50	2	Divorced	College

Clustering Houses



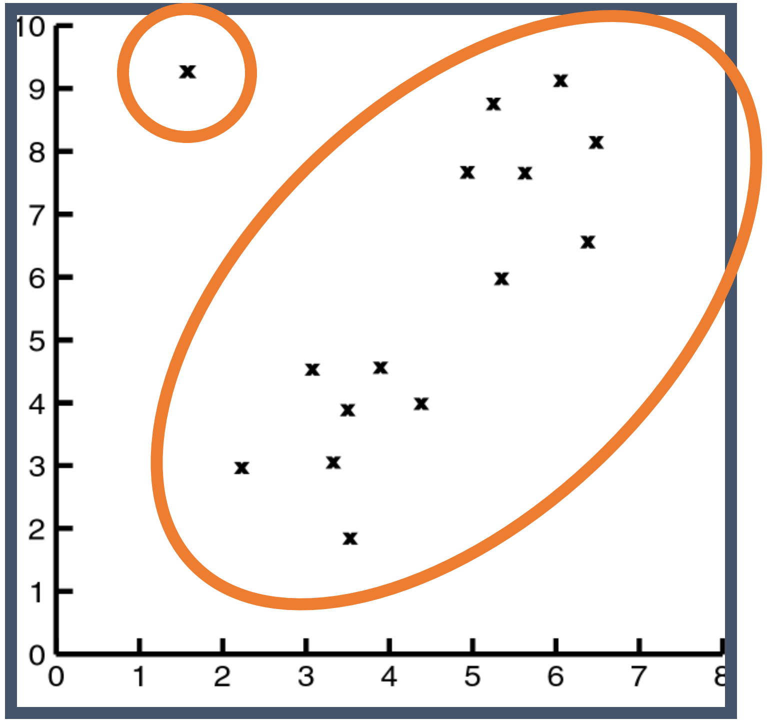
Clustering vs. Classification

- No prior knowledge
  - Number of clusters
  - Meaning of clusters
- Unsupervised learning

Clustering Issues

- Outlier handling
- Dynamic data
- Interpreting results
- Evaluating results
- Number of clusters
- Data to be used
- Scalability

Impact of Outliers on Clustering



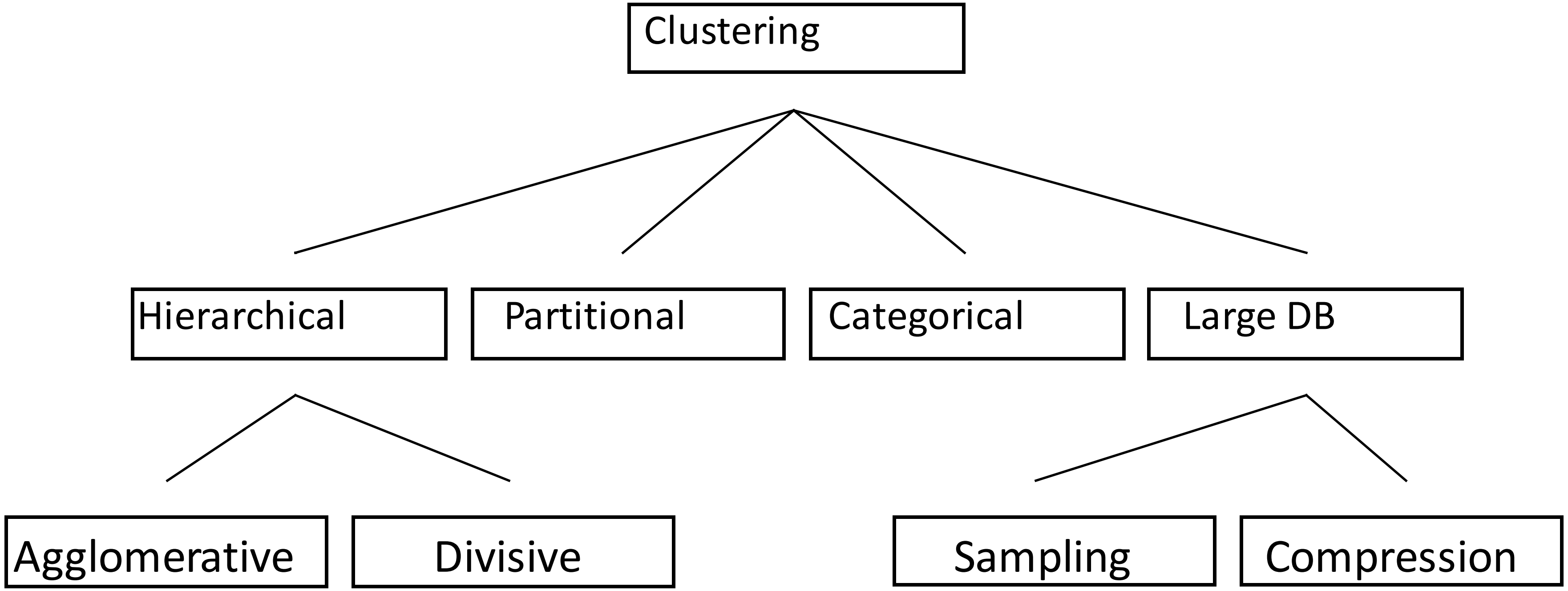
Clustering Problem

- Given a database $D=\{t_1,t_2,\ldots,t_n\}$ of tuples and an integer value k, the ***Clustering Problem*** is to define a mapping $f:D \rightarrow \{1,..,k\}$ where each t_i is assigned to one cluster K_j, $1<=j<=k$.
- A ***Cluster***, K_j, contains precisely those tuples mapped to it.
- Unlike classification problem, clusters are not known a priori.

Types of Clustering

- ***Hierarchical*** – Nested set of clusters created.
- ***Partitional*** – One set of clusters created.
- ***Incremental*** – Each element handled one at a time.
- ***Simultaneous*** – All elements handled together.
- ***Overlapping/Non-overlapping***

Clustering Approaches



Cluster Parameters

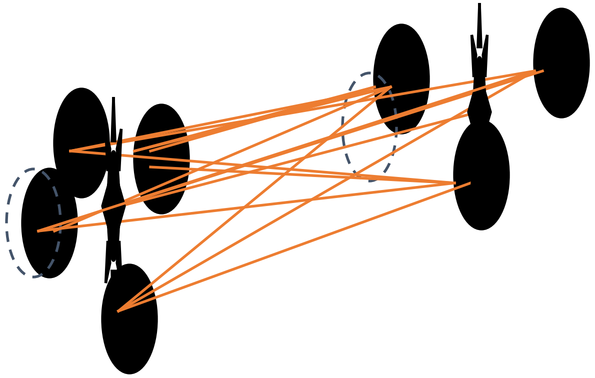
$$centroid = C_m = \frac{\sum_{i=1}^{N}(t_{mi})}{N}$$

$$radius = R_m = \sqrt{\frac{\sum_{i=1}^{N}(t_{mi} - C_m)^2}{N}}$$

$$diameter = D_m = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(t_{mi} - t_{mj})^2}{(N)(N-1)}}$$

Distance Between Clusters

- ***Single Link***: smallest distance between points
- ***Complete Link:*** largest distance between points
- ***Average Link:*** average distance between points
- ***Centroid:*** distance between centroids



Hierarchical Clustering

- Clusters are created in levels actually creating sets of clusters at each level.
- ***Agglomerative***
  - Initially each item in its own cluster
  - Iteratively clusters are merged together
  - Bottom Up
- ***Divisive***
  - Initially all items in one cluster
  - Large clusters are successively divided
  - Top Down

Hierarchical Clustering

- Clusters are created in levels actually creating sets of clusters at each level.
- ***Agglomerative***
  - Initially each item in its own cluster
  - Iteratively clusters are merged together
  - Bottom Up
- ***Divisive***
  - Initially all items in one cluster
  - Large clusters are successively divided
  - Top Down

Partitional Clustering

- Nonhierarchical
- Creates clusters in one step as opposed to several steps.
- Since only one set of clusters is output, the user normally has to input the desired number of clusters, k.
- Usually deals with static sets.

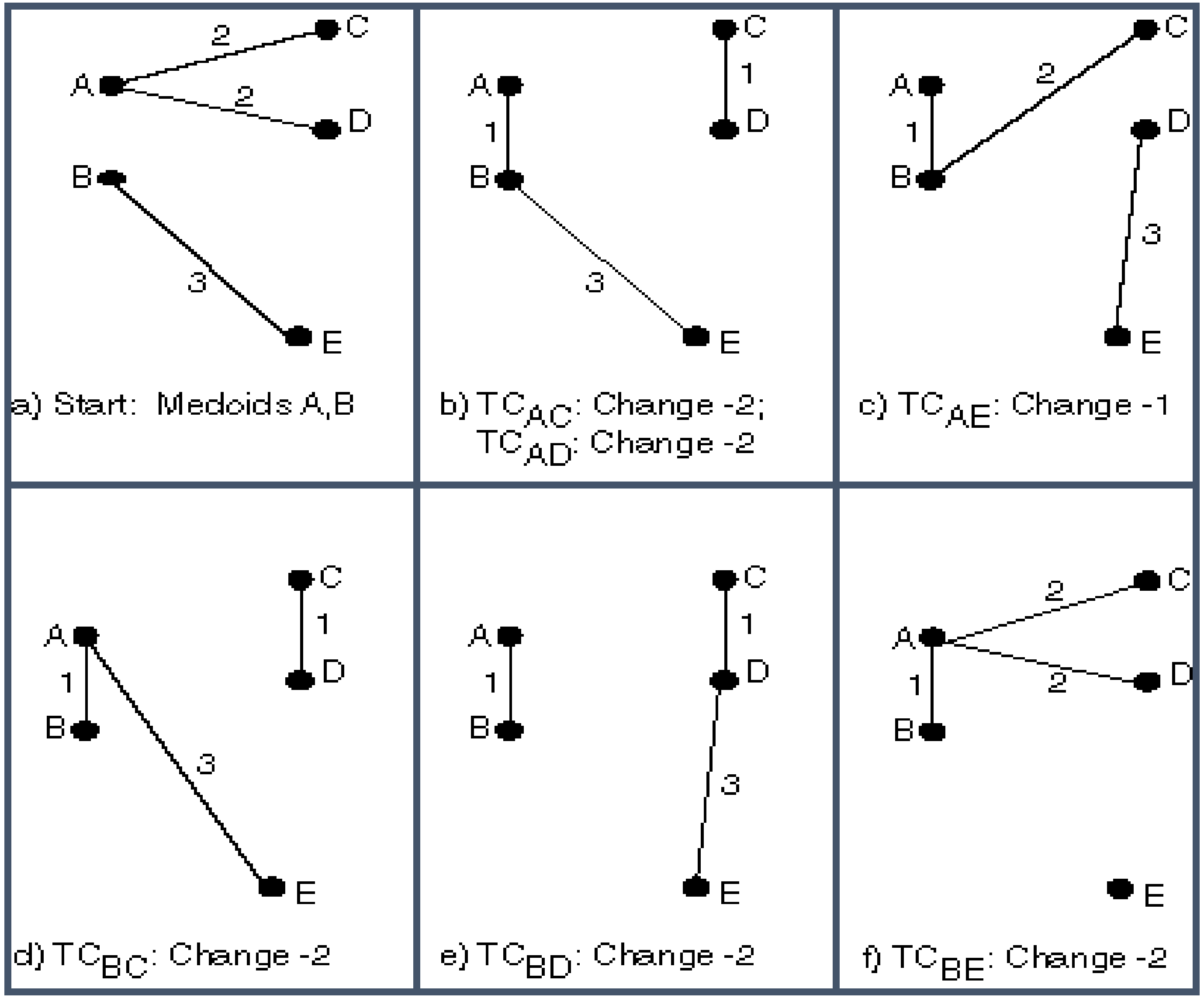
Partitional Algorithms

- MST
- Squared Error
- K-Means
- Nearest Neighbor
- PAM
- BEA
- GA

PAM

- ***Partitioning Around Medoids (PAM) (K-Medoids)***
- Handles outliers well.
- Ordering of input does not impact results.
- Does not scale well.
- Each cluster represented by one item, called the ***medoid.***
- Initial set of k medoids randomly chosen.

PAM



A
B
C
D
E
2
2
3
a) Start: Medoids A,B
A
B
C
D
E
1
1
3
b) TCAC: Change -2;
TCAD: Change -2
A
B
C
D
E
1
2
3
c) TCAE: Change -1
A
B
C
D
E
1
1
3
d) TCBC: Change -2
A
B
C
D
E
1
1
3
e) TCBD: Change -2
A
B
C
D
E
1
2
2
f) TCBE: Change -2

PAM Cost Calculation

- At each step in algorithm, medoids are changed if the overall cost is improved.
- C_{jih} – cost change for an item t_j associated with swapping medoid t_i with non-medoid t_h.

1. $t_j \in K_i$, **but** $\exists$ **another medoid** t_m *where* $dis(t_j, t_m) \leq dis(t_j, t_h)$
2. $t_j \in K_i$, **but** $dis(t_j, t_h) \leq dis(t_j, t_m) \forall$ *other medoids* t_m;
3. $t_j \in K_m, \notin K_i$, **and** $dis(t_j, t_m) \leq dis(t_j, t_h)$; **and**
4. $t_j \in K_m, \notin K_i$, **but** $dis(t_j, t_h) \leq dis(t_j, t_m)$.

PAM Algorithm

Input:

$D = \{t_1, t_2, ..., t_n\}$ // Set of elements
A // Adjacency matrix showing distance between elements.
k // Number of desired clusters.

Output:

K // Set of clusters.

PAM Algorithm:

```
arbitrarily select k medoids from D;
repeat
    for each t_h not a medoid do
        for each medoid t_i do
            calculate TC_ih;
    find i,h where TC_ih is the smallest;
    if TC_ih < 0 then
        replace medoid t_i with t_h;
until TC_ih >= 0;
for each t_i ∈ D do
    assign t_i to K_j where dis(t_i, t_j) is the smallest over all medoids;
```

2.Density-based methods

To discover **clusters with arbitrary shape**, density-based clustering methods have been developed. These typically regard clusters as dense regions of objects in the data space which are separated by regions of low density (representing noise).

DBSCAN: A density-based clustering method based on connected regions with sufficiently high density

DBSCAN is a density-based clustering algorithm. The algorithm grows regions with sufficiently high density into clusters, and discovers clusters of arbitrary shape in spatial databases with noise. It defines a cluster as a maximal set of density-connected points.

The basic ideas of density-based clustering involve a number of new definitions. The neighborhood within a radius **ε** of a given object is called the **ε** -neighborhood of the object.

- The neighborhood within a radius ε of a given object is called the **ε-neighborhood** of the object.
- If the ε-neighborhood of an object contains at least a minimum number, *MinPts*, of objects, then the object is called a **core object.**
- Given a set of objects, D, we say that an object p is **directly density-reachable** from object q if p is within the ε-neighborhood of q, and q is a core object.

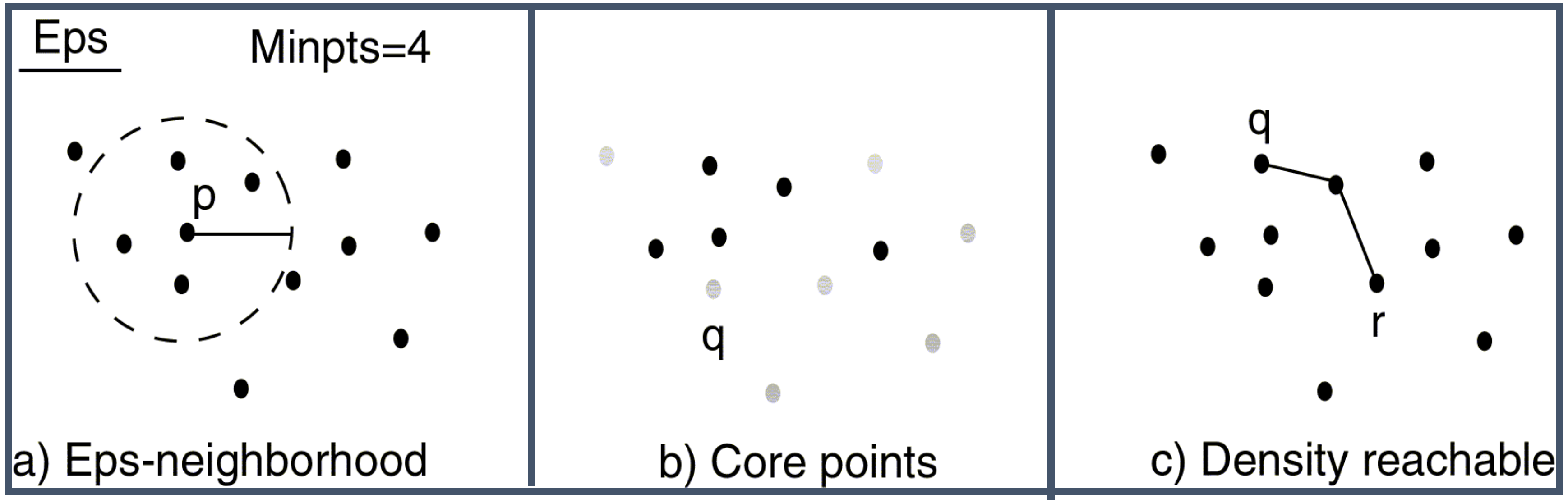
DBSCAN

- Density Based Spatial Clustering of Applications with Noise
- Outliers will not effect creation of cluster.
- Input
  - ***MinPts*** – minimum number of points in cluster
  - ***Eps*** – for each point in cluster there must be another point in it less than this distance away.

DBSCAN Density Concepts

- ***Eps-neighborhood:*** Points within Eps distance of a point.
- ***Core point:*** Eps-neighborhood dense enough (MinPts)
- ***Directly density-reachable:*** A point p is directly density-reachable from a point q if the distance is small (Eps) and q is a core point.
- ***Density-reachable:*** A point si density-reachable form another point if there is a path from one to the other consisting of only core points.

Density Concepts



DBSCAN Algorithm

Input:

$D = \{t_1, t_2, ..., t_n\}$ **//Set of elements.**

$MinPts$ **// Number of points in cluster.**

Eps **// Maximum distance for density measure.**

Output:

$K = \{K_1, K_2, ..., K_k\}$ **//Set of clusters.**

DBSCAN Algorithm:

$k = 0$; **// Initially there are no clusters.**

for $i = 1\ to\ n$ **do**

  if t_i **is not in a cluster then**

    $X = \{t_j \mid t_j$ **is density-reachable from** $t_i\}$;

    if X **is a valid cluster then**

      $k = k + 1$;

      $K_k = X$;

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- $MinPts$: the neighborhood density threshold.

Output: A set of density-based clusters.

Method:

(1) mark all objects as unvisited;
(2) **do**
(3) randomly select an unvisited object $\boldsymbol{p}$;
(4) mark $\boldsymbol{p}$ as visited;
(5) **if** the ϵ-neighborhood of $\boldsymbol{p}$ has at least $MinPts$ objects
(6) create a new cluster C, and add $\boldsymbol{p}$ to C;
(7) let N be the set of objects in the ϵ-neighborhood of $\boldsymbol{p}$;
(8) **for** each point $\boldsymbol{p'}$ in N
(9) if $\boldsymbol{p'}$ is unvisited
(10) mark $\boldsymbol{p'}$ as visited;
(11) if the ϵ-neighborhood of $\boldsymbol{p'}$ has at least $MinPts$ points, add those points to N;
(12) if $\boldsymbol{p'}$ is not yet a member of any cluster, add $\boldsymbol{p'}$ to C;
(13) **end for**
(14) output C;
(15) **else** mark $\boldsymbol{p}$ as noise;
(16) **until** no object is unvisited;

DBSCAN algorithm