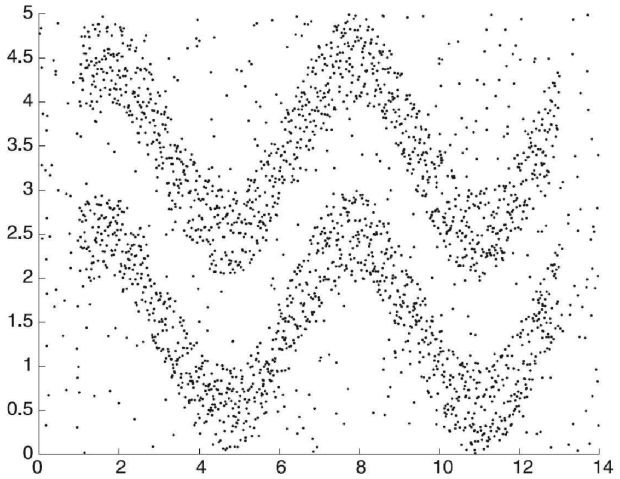
Clustering
Grid & Density-based Algorithms

Object space is divided into finite number of cells that form grid like structure.

Why is complexity of grid based algorithm O(n2) in worst case and O(nlogn)?

Why Grid-Density-Based ?

- Representative-Based clustering Algorithms are also distance based.
- One of the major problems is that the shape of the underlying clusters is already defined implicitly by the underlying distance function.
- For example, a *k*-means algorithm implicitly assumes a spherical shape for the cluster.
- In practice, the clusters may be hard to model with a prototypical shape implied by a distance function.

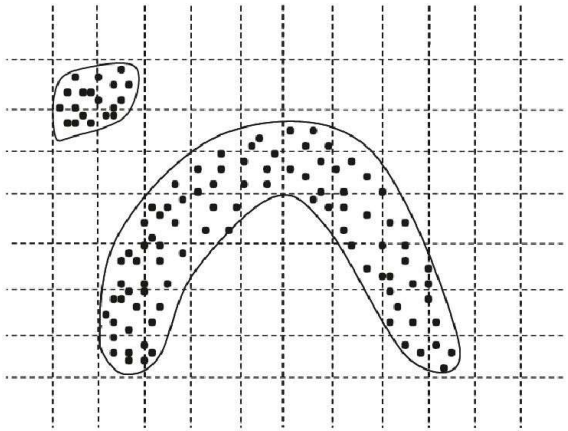


Grid & Density-based Algorithms

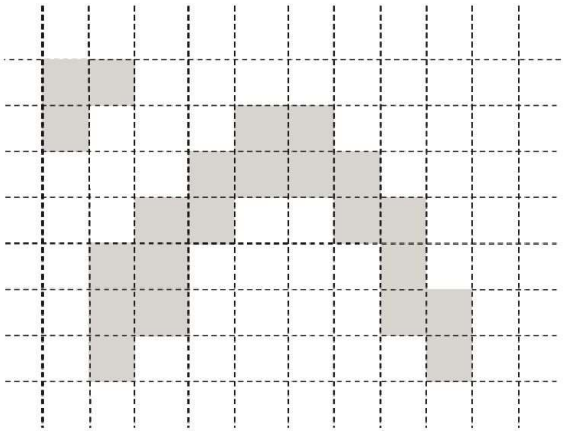
- The core idea in density-based algorithms is to perform two iterations:
    1) detect dense areas in the data,
    2) then grow and merge them.
- It is thus possible to detect clusters with complex shapes.
- Many variations of this broader principle exist.
    - DBSCAN: pre-selected data points in dense regions are clustered with a single-linkage method
    - DENCLUE: use gradient ascent on the kernel-density estimates

Grid-Based Algorithms

- Data is discretized into p intervals, typically equi-width
- A total of p^d hyper-cubes, where d is the number of dimensions in the underlying data
- Clustering is done over the hyper-cubes
- A density threshold τ is used to determine the subset of the p^d hyper-cubes that are dense



(a) Data points and grid



(b) Agglomerating adjacent grids

Grid-Based Algorithms

- *Adjacently connected*: if two cells a side/face
  - A weaker version: if they share a *corner* in common.
- *Density connected*, if a path can be found from one cell to the other, via a sequence of *adjacently connected*
- Grid-Based Clustering maps to finding the *connected regions* created in such grid
- Easy to implement as graph
  - *dense* grid cells are graph nodes
  - edges represents *adjacent connectivity*
  - => connected components are clusters
- Breadth-first or depth-first traversal on the graph for finding connected components
- What is the complexity ?

Grid-Based Algorithms

Algorithm *GenericGrid*(Data: $\mathcal{D}$, Ranges: p, Density: τ)
begin
  Discretize each dimension of data $\mathcal{D}$ into p ranges;
  Determine dense grid cells at density level τ;
  Create graph in which dense grids are connected if they are adjacent;
  Determine connected components of graph;
  return points in each connected component as a cluster;
end

Parameters

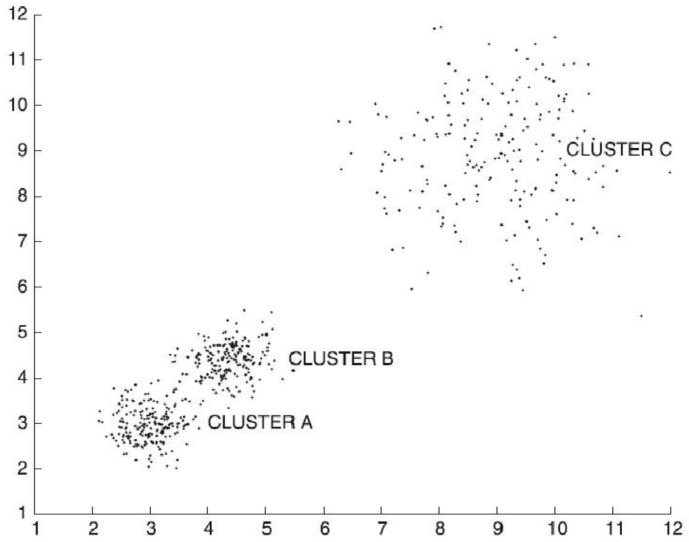
- Number of data clusters is not predefined, in contrast to representative-based
- On the other hand, two different parameters need to be defined:
  - Grid resolution, number of intervals p
  - Density threshold τ .

Choosing p & τ

- When p is too small ?
  - points from multiple clusters will be present in the same grid region => undesirable merging of clusters.
- When p is too large ?
  - Many empty grid cells => natural clusters in the data may be disconnected
- When τ is too low ?
  - clusters including the ambient noise, will be merged
- When τ is too high ?
  - We can partially or entirely miss a cluster.

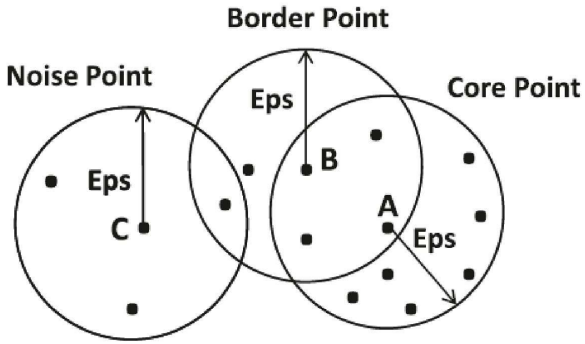
Practical Issues

- A major issue is the use a single density parameter τ
- The grid grows quickly with increasing dimensionality, becoming computationally infeasible in high dimensions



For a database of size n, the time complexity can be O(n2) in the worst case. However, for some special cases, the use of a spatial index for finding the nearest neighbors can reduce this to approximately O(n·log(n)) distance computations

DBSCAN



- Widely used clustering Algorithm.
- Given a set of data points, and two parameters *Eps*, τ, DBSCAN is able to distinguish three types of points:
  - **Core points** are the ones that have at least τ points in their Eps-neighborhood. This is the circle centered at the point itself and has a radius of Eps.
  - **Border points** are the points that fall into the neighborhood of some core points, but themselves are not core points. These points are typically located close to the cluster border, hence their name. They belong to some dense cluster, but themselves are not in the middle of a dense area. Only some side of their neighborhood is dense, but not their complete neighborhood.
  - **Noise points** are the points that are neither core nor border. Basically noise points fall in sparse areas, where the neighborhood of the point itself is not dense, and no core points exist around in the neighborhood.

DBSCAN from Mars

Algorithm *DBSCAN*(Data: $\mathcal{D}$, Radius: Eps, Density: τ)
begin
  Determine core, border and noise points of $\mathcal{D}$ at level (Eps, τ);
  Create graph in which core points are connected
    if they are within Eps of one another;
  Determine connected components in graph;
  Assign each border point to connected component
    with which it is best connected;
  return points in each connected component as a cluster;
end

DBSCAN - Reachability

- (p_i, p_j) are *directly density reachable* if p_i is a core point, and p_j is in the Eps-neighborhood of p_i. Note that this is a non-symmetric relation.
- (p_i, p_j) are *density reachable* if p_i is a core point, and their is a chain of core points P_{i+1} ... p_n, where every consecutive pair in this chain is directly density reachable, and p_j is in the Eps-neighborhood of p_n. In this case, p_j is a border point.
- (p_i, p_j) are *density connected* if both p_i and p_j are density reachable from some point p_k. In this case, both p_i, p_j are border point. This relation is symmetric.

- A cluster is a maximal set of points that are all reachable from one another under any of these three definitions of reachability. A point that is neither directly density reachable, nor density reachable from any core point is a noise that does not belong to any cluster.

```
DBSCAN(D, eps, MinPts)
  C = 0
  for each unvisited point P in dataset D
    NeighborPts = regionQuery(P, eps)
    if sizeof(NeighborPts) < MinPts
        mark P as visited
    else
        C = next cluster
        expandCluster(P, NeighborPts, C, eps, MinPts)

expandCluster(P, NeighborPts, C, eps, MinPts)
  mark P as visited
  add P to cluster C
  for each point P' in NeighborPts
    if P' is not visited
        mark P' as visited
        NeighborPts' = regionQuery(P', eps)
        if sizeof(NeighborPts') >= MinPts
            NeighborPts = NeighborPts joined with NeighborPts'
        if P' is not yet member of any cluster
            add P' to cluster C
```

Time Complexity: O(nlogn) or O(n2)

What is the complexity of DBSCAN?

- The major time complexity of *DBSCAN* is in finding the neighbors of the different data points within a distance of *Eps*.
- For a database of size n, the time complexity can be $O(n^2)$ in the worst case.
- The use of a spatial index for finding the nearest neighbors can reduce this to approximately $O(n \cdot \log(n))$ distance computations.

Advantages
1. Used to deal with clusters of arbitrary shape unlike representative based cluster
2. is great with handling outliers within dataset
3. good at separating clusters of high density versus clusters of low density within a dataset

Disadvantages
1. Determining value of p and T is difficult
2. Not useful when dealing with clusters of varying densities

Practical Issues

- *Eps*, τ are related to one another, how ?
- Progressive DBSCAN:
  - Using the same value for τ, apply DBSCAN in a progressive way, with increasing values of Eps.
  - Start with a small Eps to find intersections are rather dense, then
  - iteratively relax the eps value to find less dense clusters.
  - After every iteration, the points that already belong to some cluster are removed.

Map-making example
Building road segments and detecting turns from gps tracks
M Ezzat, M Sakr, R Elgohary, ME Khalifa - Journal of computational science, 2018

DENCLUE

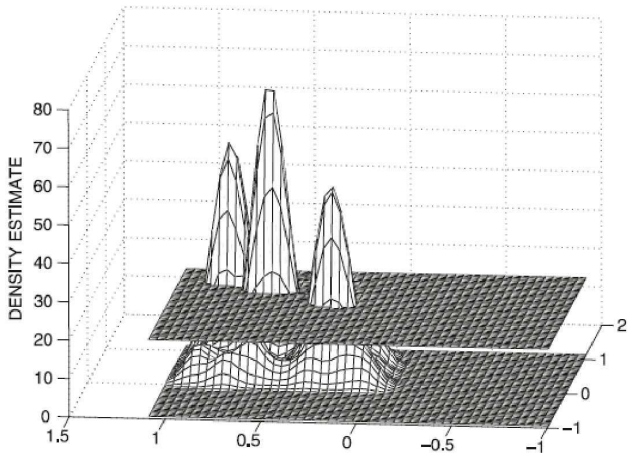
- The *DENCLUE* algorithm is based on the statistical foundations of kernel density estimation.
- Kernel-density estimation creates a smooth profile of the density distribution.
- In kernel-density estimation, the density $f(X)$ at coordinate X is defined as a sum of the influence (kernel) functions $K(\cdot)$ over the n different data points in the database D:
- For a Gaussian kernel

$$f(\overline{X}) = \frac{1}{n} \sum_{i=1}^{n} K(\overline{X} - \overline{X_i})$$

$$K(\overline{X} - \overline{X_i}) = \left(\frac{1}{h\sqrt{2\pi}} \right)^d e^{-\frac{||\overline{X} - \overline{X_i}||^2}{2 \cdot h^2}}$$

DENCLUE

- Intuitively, the effect of kernel-density estimation is to replace each discrete data point with a smooth “bump,” and the density at a point is the sum of these “bumps.”
- This results in a smooth profile of the data in which the random artifacts of the data are suppressed, and a smooth estimate of the density is obtained.
- *h* represents the bandwidth of the estimation that regulates the smoothness of the estimation. Large values of the bandwidth *h* smooth out the noisy artifacts but may also lose some detail about the distribution.
- The goal is to determine clusters by using a density threshold τ that intersects this smooth density profile.



Clustering Methods

- Representative-Based Algorithms
  - K-Means, K-Medians, K-Medoids.
  - Find a high quality set of representatives, then link data points to their closest representatives.
- Hierarchical Clustering Algorithms
  - Agglomerative: keep grouping similar objects/nodes.
  - Divisive: keep splitting dissimilar groups.
- Probabilistic Model-Based Algorithms:
  - Soft clustering.
- Density-Based Algorithms
  - DBSCAN, DENCLUE.