Density-Based Clustering

Introduction

Partitioning methods (**K-means**) and **hierarchical clustering**

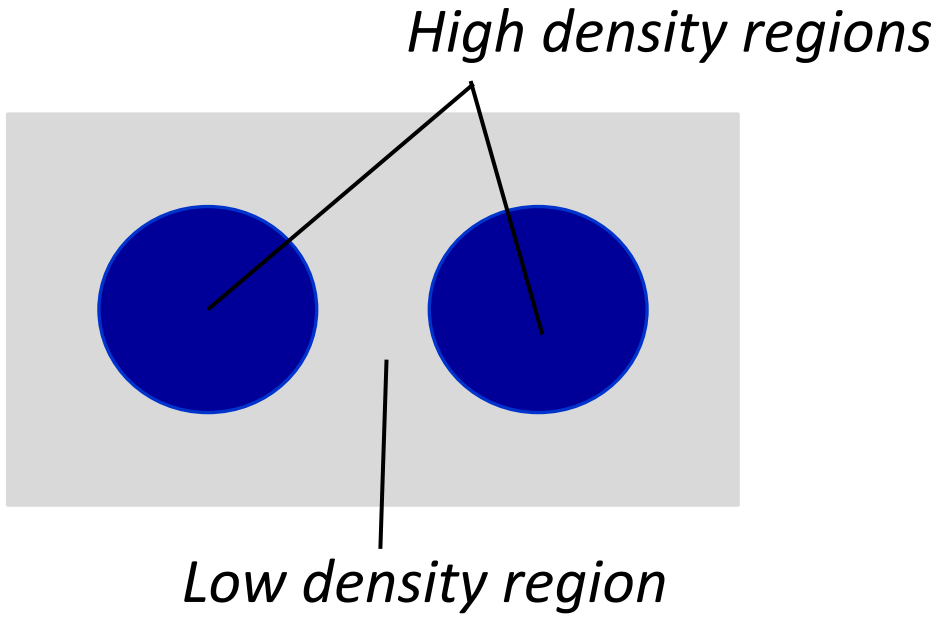
- Suitable for finding spherical-shaped clusters
- They are also severely affected by the presence of noise and outliers in the data

Density based clustering

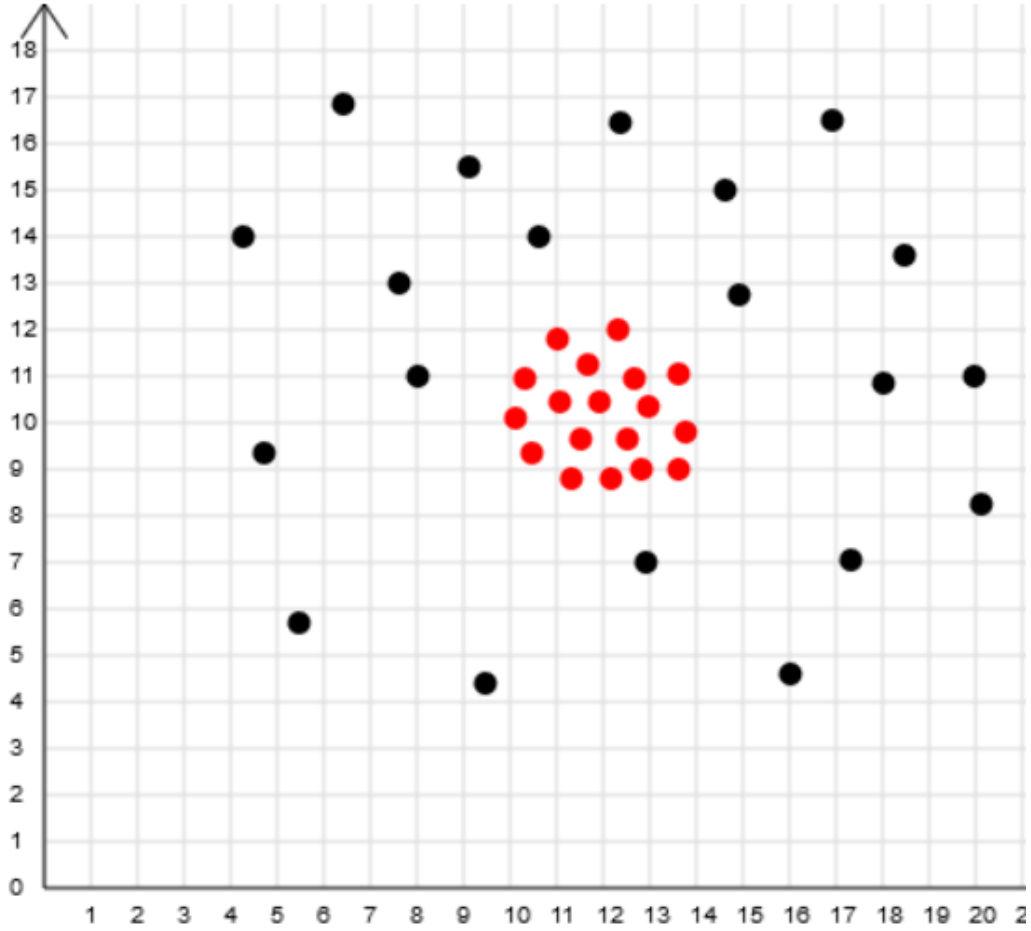
- identify clusters of any shape in data set containing noise and outliers

Definition

- A cluster is a dense region of objects surrounded by a region of low density

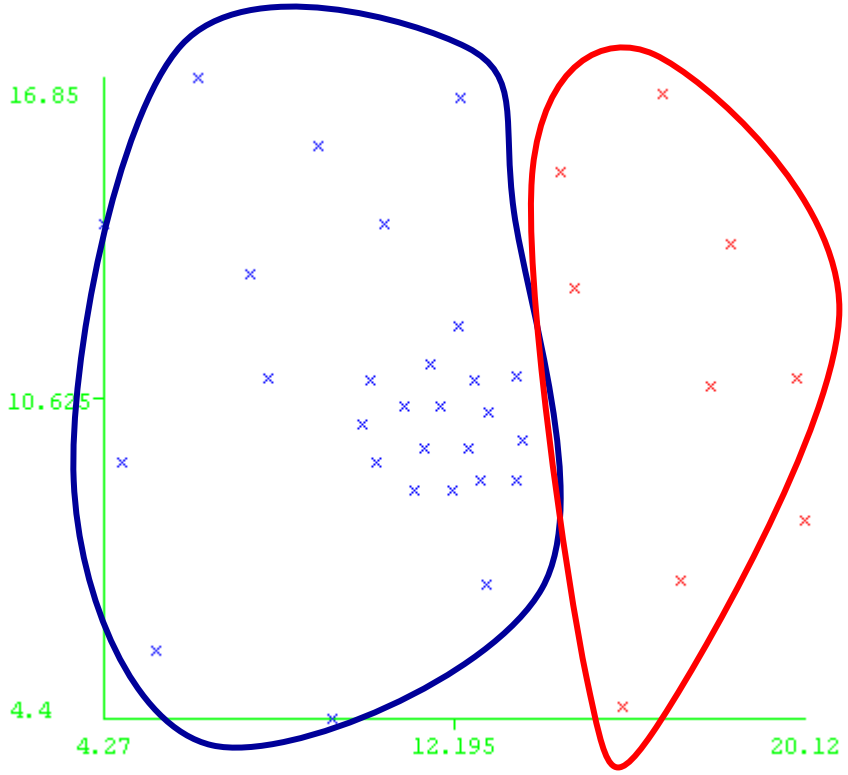


Example

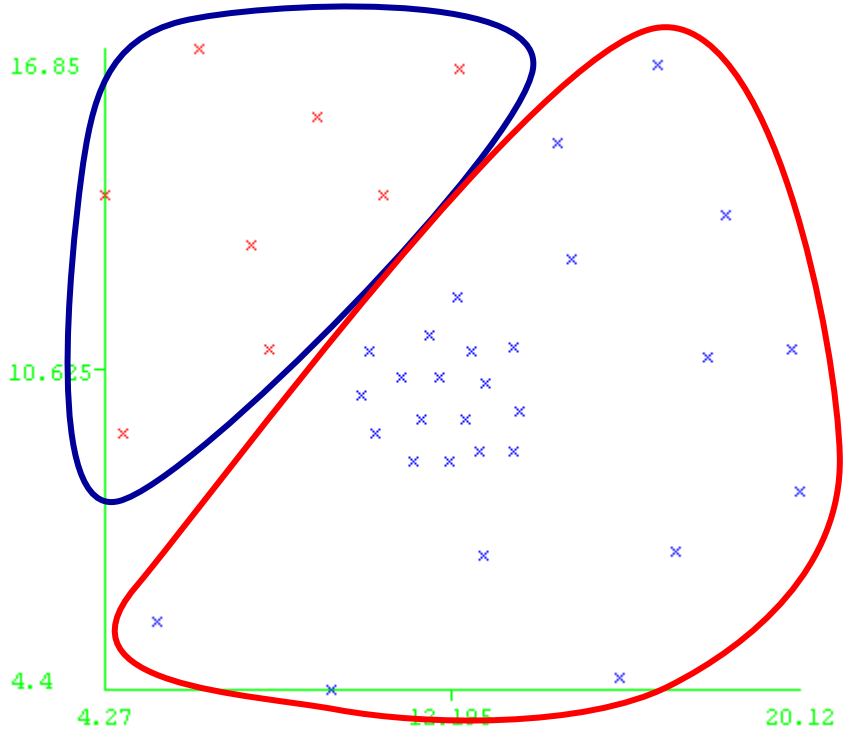


The data set contains a high density cluster surrounded by a low density cluster

K-Means Result



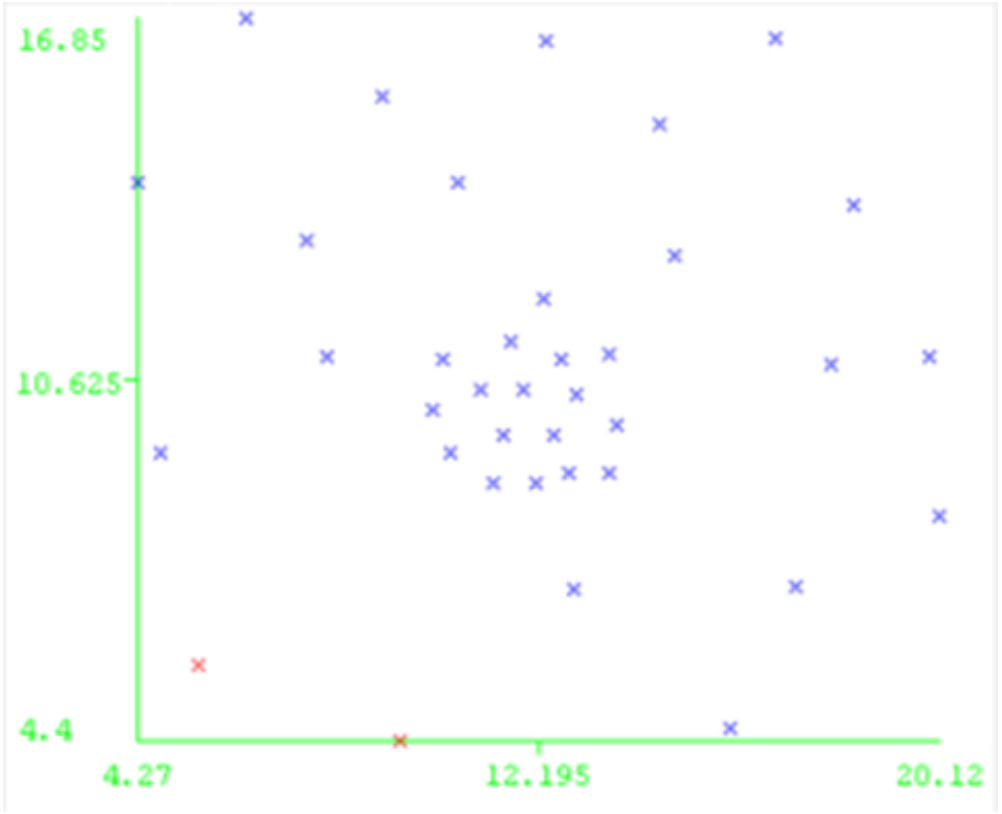
Random centroids run 1



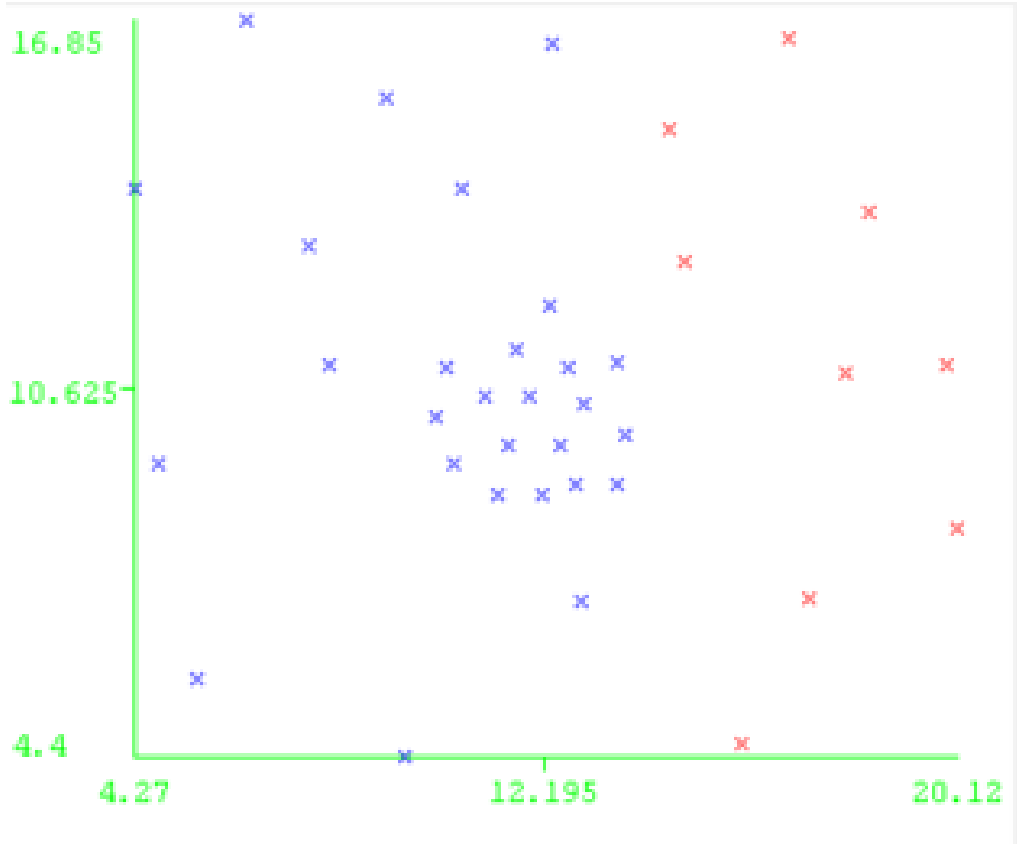
Random centroids run 2

K-means does not find the natural clusters in the data

Hierarchical Clustering Result

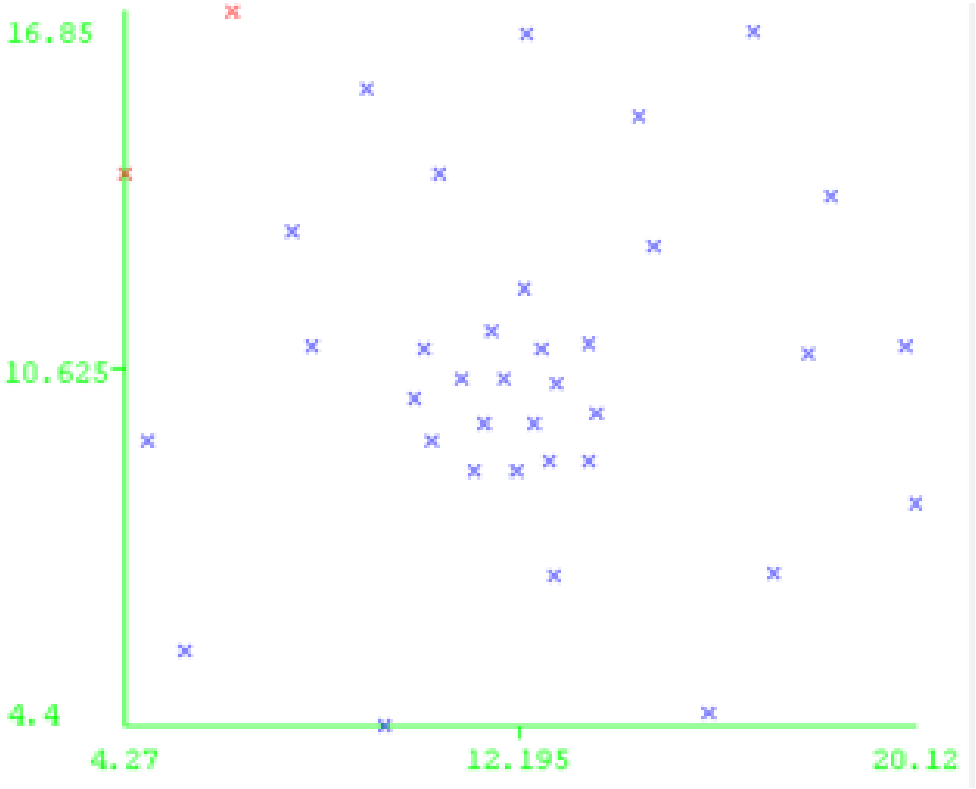


Single Link (Min)

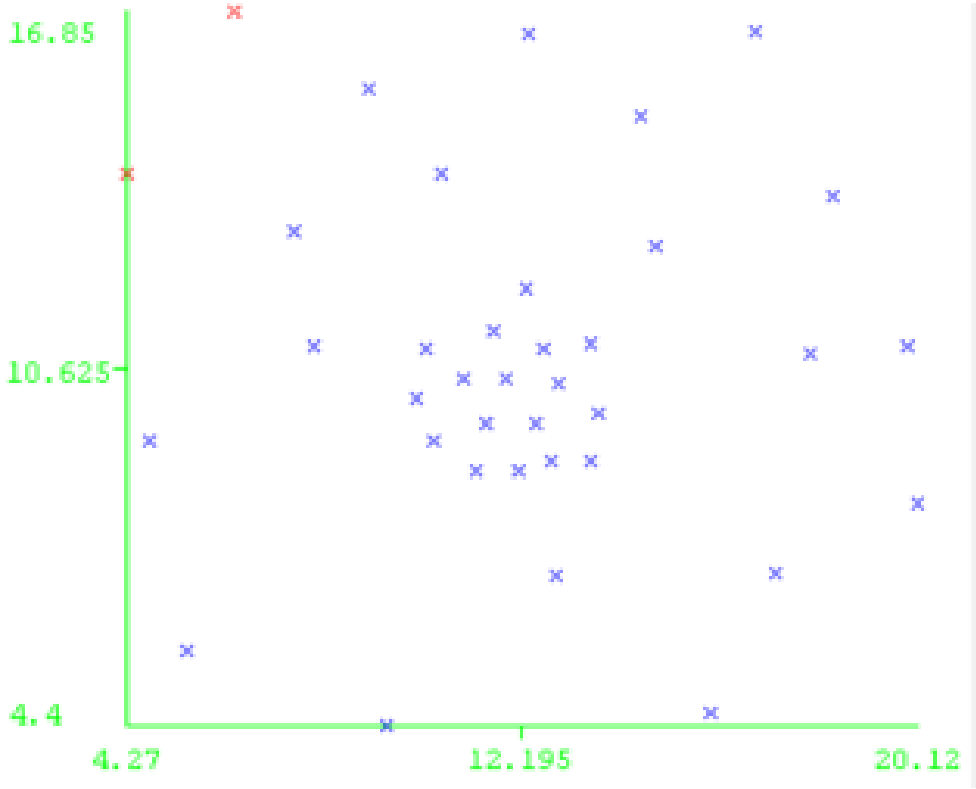


Complete Link (Max)

Hierarchical Clustering Result



Centroid



Group Average

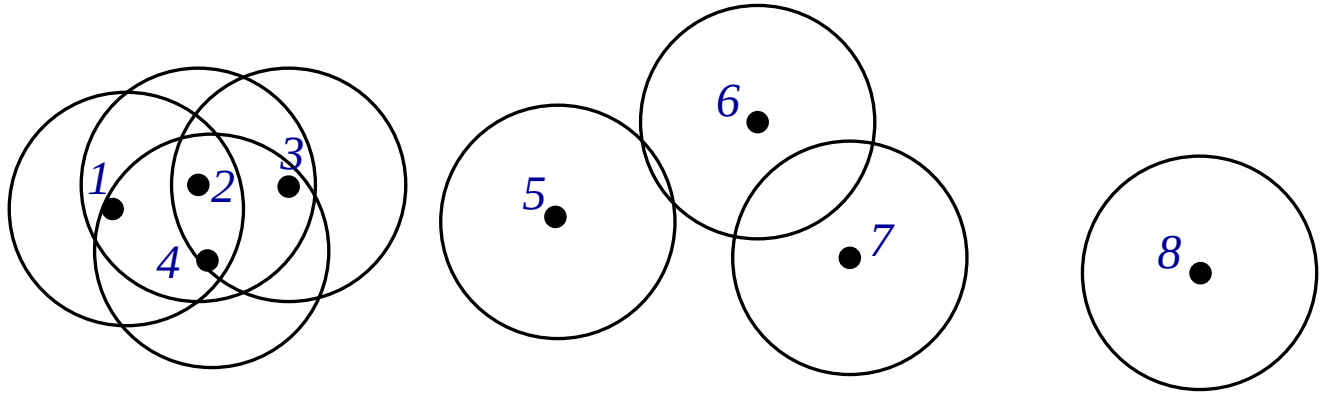
Density Based Approaches

- **Center-Based approach**
  - density is estimated for a particular point in the dataset by counting the number of points within a particular radius
  - data driven approach

- **Grid Based approach**
  - Partition the whole space into cells with grids and then merge the cells to build clusters.
  - data and space driven approach

Center Based Approaches

- Need a density measure
- Center-based approaches:
  - Density for particular point is the number of points within a specified radius, including the point itself



Point	Density
1	3
2	4
3	3
4	4
5	1
6	1
7	1
8	1

Center Based

- Density depends on the given radius
- How does radius affect the density?

Density increases with radius

- If radius is large enough: all points will have the same density, n
- If radius is too small: all points will have density of 1

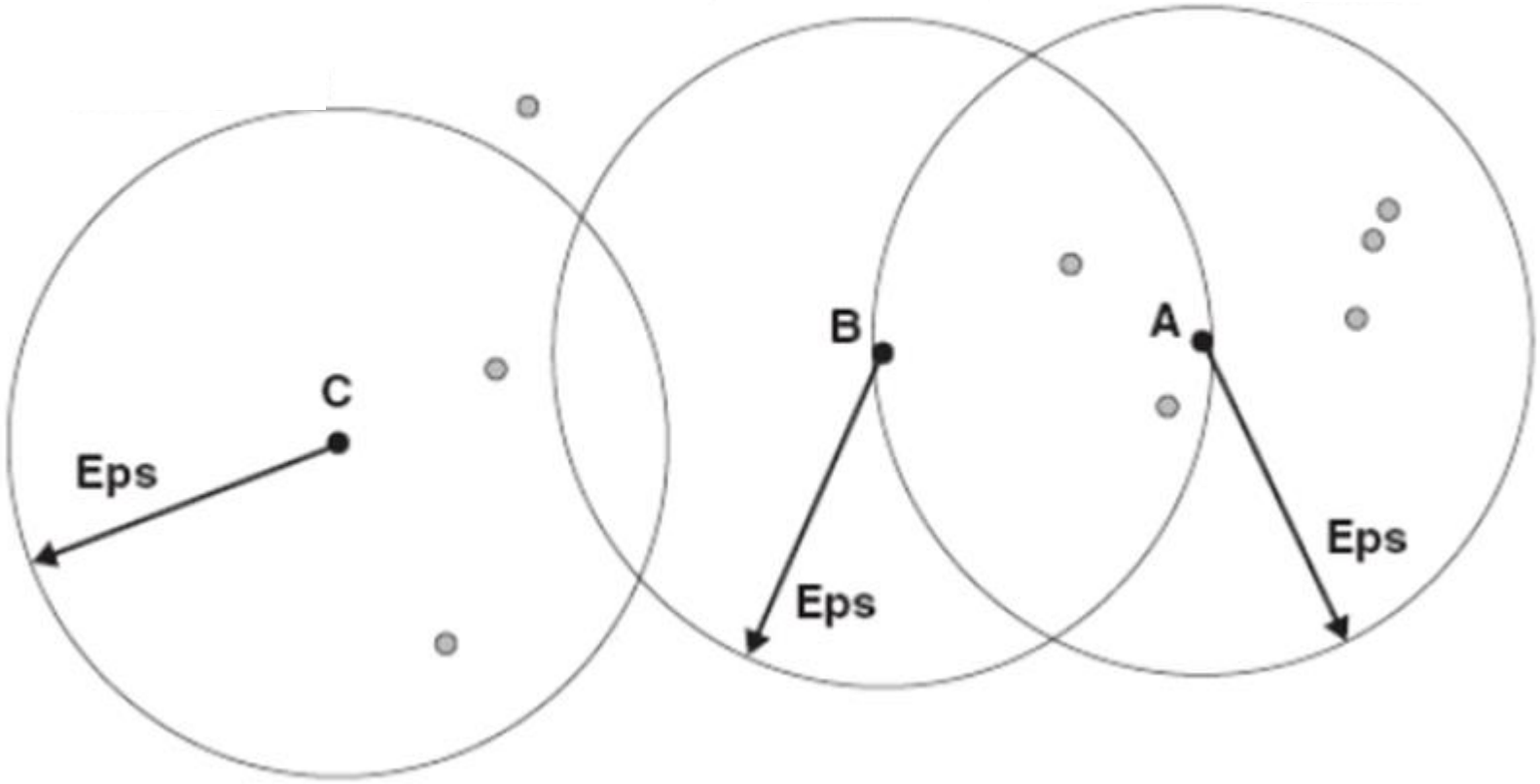
Center Based

- The center-based approach classifies a point as being:

  - **Core point**: if its density within a given *Radius (eps)* exceeds or equal to a certain threshold *MinPts*
    - The interior of a dense region
  - **Border point**: not core point, but falls in the neighborhood of a core point
    - On the edge of a dense region
  - **Noise point**: any point that is not a core point nor a border point

Center Based

MinPts = 5



Point	Density
A	7
B	4
C	3

DBSCAN Algorithm

Given the previous definitions of points, we introduce DBSCAN algorithm
one of the most popular algorithm for density-based clustering analysis

Algorithm 8.4 DBSCAN algorithm.

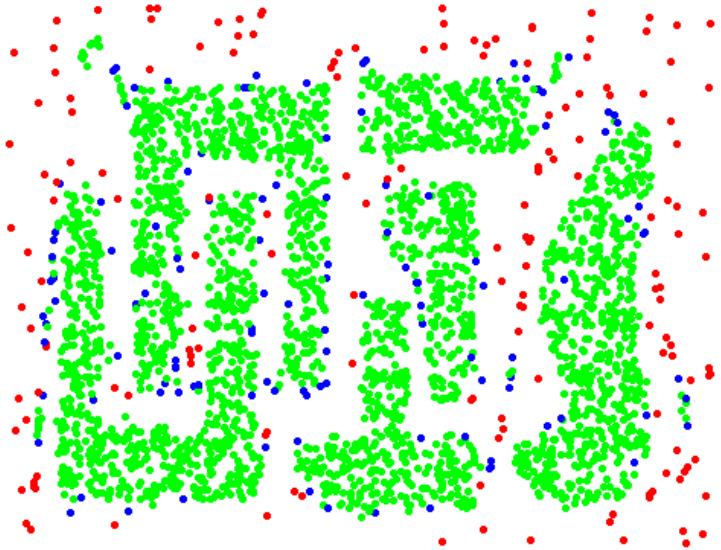
1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.

- Any two core points that are close enough, within a distance Eps of one another, are put in the same cluster
- Any border point that is close enough to a core point is put in the same cluster as that core point.

DBSCAN: Core, Border and Noise Points

Eps = 10, MinPts = 4

Original Points



Point types: core, border and noise

DBSCAN

DBSCAN Animation at https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Time and Space Complexity

- If data set consists of n points:
  - Time needed: O(n*$time\ to\ find\ points\ within\ radius\ Eps$)
  - In the worst case: O(n^2)
  - May be optimized to ($nlog(n)$) in low dimensions
  - Space needed: O(n)
    - Only keep small amount of information about each point (point type and cluster label)

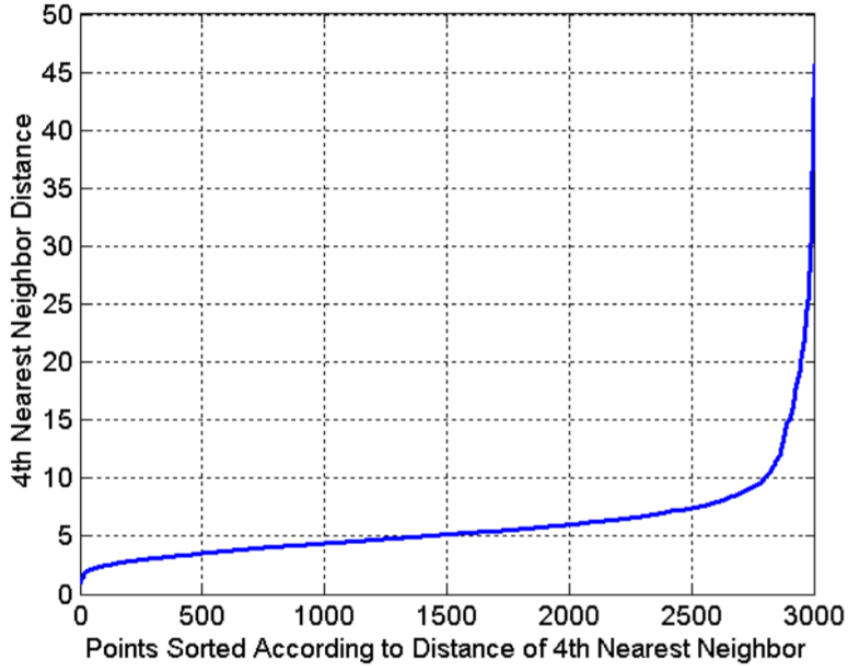
Parameter Selection

- Unlike to **K-means**, density-based does not require the user to specify the number of clusters to be generated
- Eps and MinPts – How to select?

- The basic approach is to look at the behavior of the distance from a point to its k nearest neighbor
  - K-dist: the distance from each point to its k^{th} nearest neighbor
  - If points belong to the same cluster: k-dist is small
  - For noise points: k-dist is large

Parameter Selection

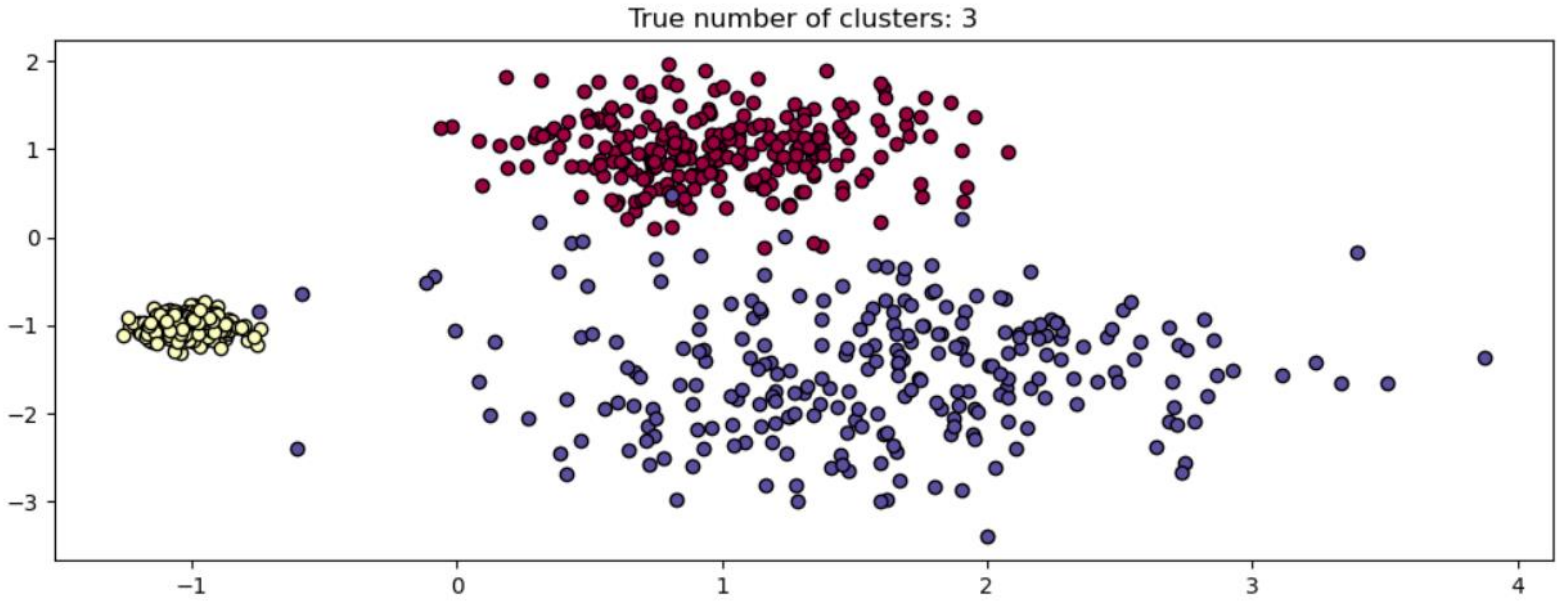
- if we compute the k-dist for all the data points for some k, sort them in increasing order, and then plot them, we expect to see a sharp change at the value of k-dist that could be a suitable Eps.



Basic Implementation

Generate Data

```
centers = [[1, 1], [-1, -1], [1.5, -1.5]]
X, labels_true = make_blobs(
    n_samples=750, centers=centers, cluster_std=[0.4, 0.1, 0.75], random_state=0
)
plot(X, labels=labels_true, ground_truth=True)
```

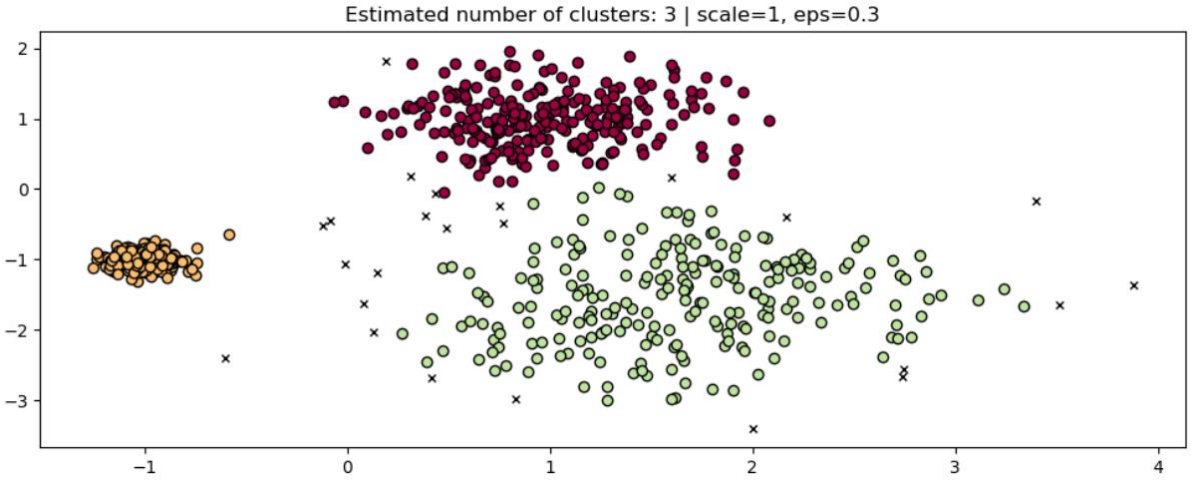


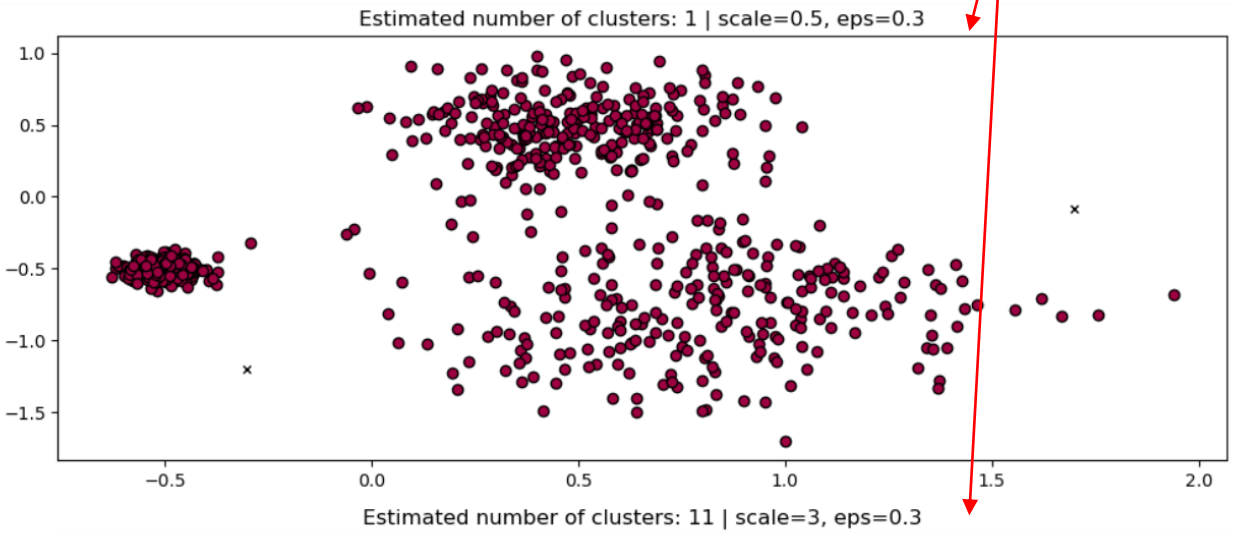
Basic Implementation (cont.)

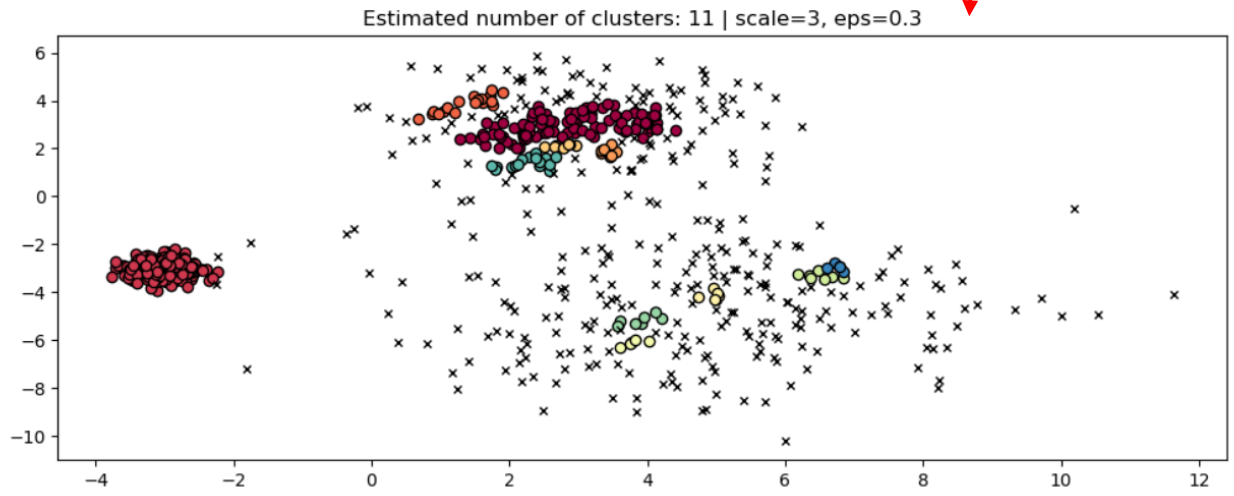
But fails when applied to rescaled versions of the dataset

```
fig, axes = plt.subplots(3, 1, figsize=(10, 12))
dbs = DBSCAN(eps=0.3)
for idx, scale in enumerate([1, 0.5, 3]):
    dbs.fit(X * scale)
    plot(X * scale, dbs.labels_, parameters={"scale": scale,
"eps": 0.3}, ax=axes[idx])
```

A eps value (0.3) works for this dataset







Parameter Selection

For a given k:

1. Compute the k-dist for all points

2. Sort them in increasing order

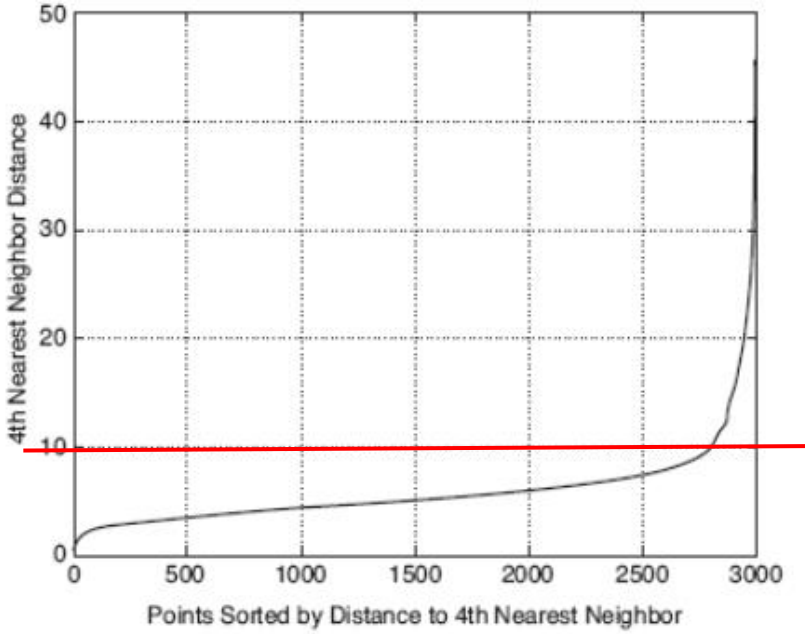
3. Plot the sorted values

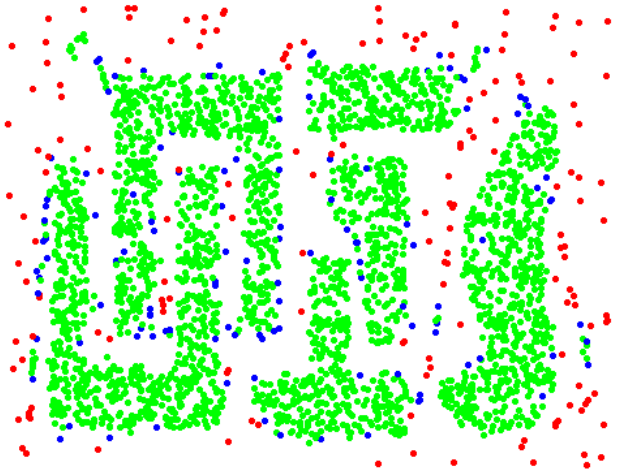
A sharp change at the value of k-dist that corresponds to a suitable Eps value

4. Select this distance as Eps

5. Select k as MinPts

Parameter Selection





Sample data (3000 pts)

K-dist plot

MinPts = 4

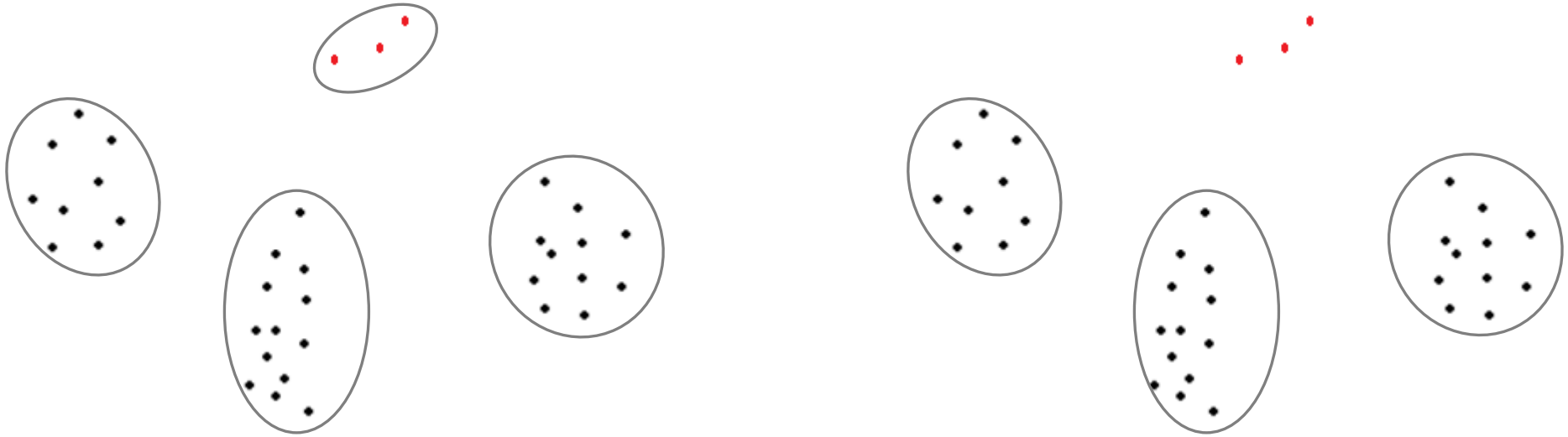
Eps = 10

Parameter Selection

The value selected for Eps depends on K (minPts)

If k is too small: noise may be treated as core clusters

If k is too large: small clusters may be labeled as noise



Parameter Selection

- There is no automatic way to determine the MinPts value for DBSCAN.
- Ultimately, the MinPts value should be set using domain knowledge and familiarity with the data set.

- From experience, here are a few rules of thumb for selecting the MinPts value:

- The larger the data set, the larger the value of MinPts should be
- If the data set is noisier, choose a larger value of MinPts
- Generally, MinPts should be greater than or equal to the dimensionality of the data set

If your data has d dimensions, choose MinPts = 2*dim

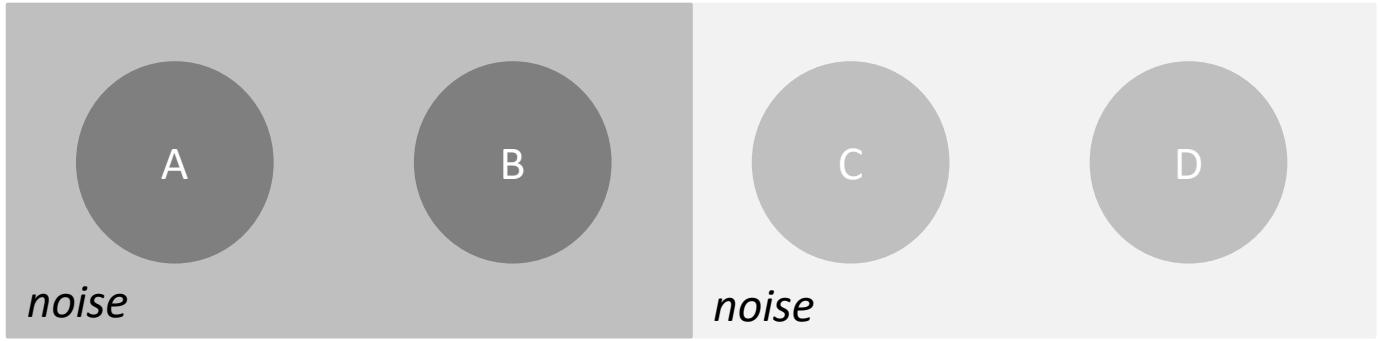
For 2-dimensional data, we use MinPts = 4.

Characteristics

- Resistant to noise
- Can handle clusters of arbitrary shapes and sizes
- Trouble with high dimensional data because density is difficult to define
- Expensive: requires computation of all pairwise proximities

Characteristics

- When densities vary widely, may not find suitable clusters



Reduce density threshold so C and D are found as clusters

➡ *Clusters A and B and noise surrounding them will be found as one cluster*

Increase density threshold so clusters A and B are found and surrounding noise is dropped

➡ *Clusters C and D will be considered noise*

Characteristics

- Produces partitional clustering
- Number of clusters is determined by the algorithm
- Points in low density regions are eliminated

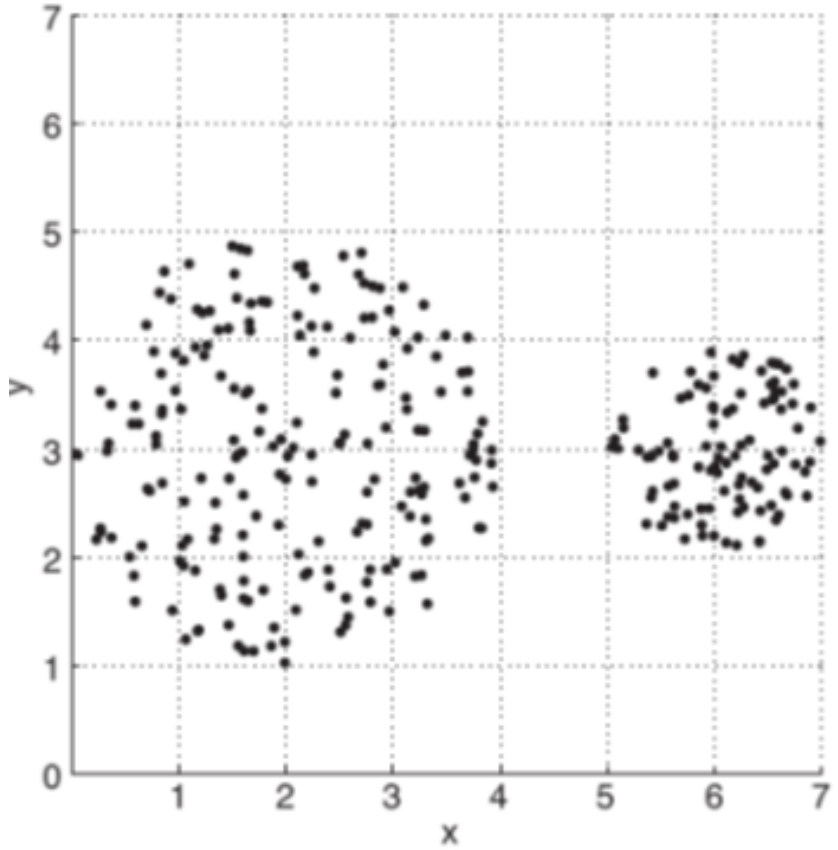
  => Partial clustering

Grid-Based Density Clustering

- Split each attribute into a number of contiguous intervals
  - Assumption: attributes are ordinal, interval or continuous

1. Define a set of grid cells
2. Assign each object to the appropriate cell and compute the density of each cell
3. Eliminate cells having densities below a certain threshold T
4. Form clusters from contiguous groups of cells

Example



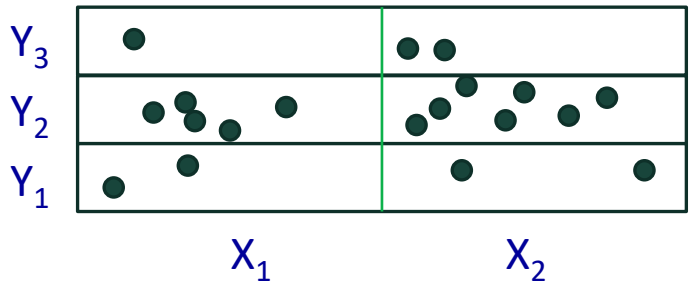
Grid-based assignment

0	0	0	0	0	0	0
0	0	0	0	0	0	0
4	17	18	6	0	0	0
14	14	13	13	0	18	27
11	18	10	21	0	24	31
3	20	14	4	0	0	0
0	0	0	0	0	0	0

Point counts for each cell

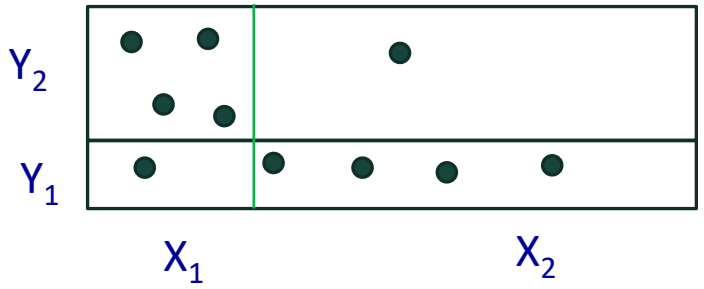
Defining grid cells

- Many ways to split attribute values into a number of contiguous intervals
  - Split into equal width intervals



Cells have the same volume

  - Split into equal frequency
  - Use clustering



Cells may have different volumes

Density of a cell

- Natural definition: number of points divided by the volume (per amount of space)
  - Number of road signs per mile (1-dimensional)
  - Number of eagles per square kilometer of habitat (2-dimensional)
  - Number of molecules per cubic centimeter (3-dimensional)

- If all cells have the same volume:
  - Number of points per space is equivalent to number of points per cell

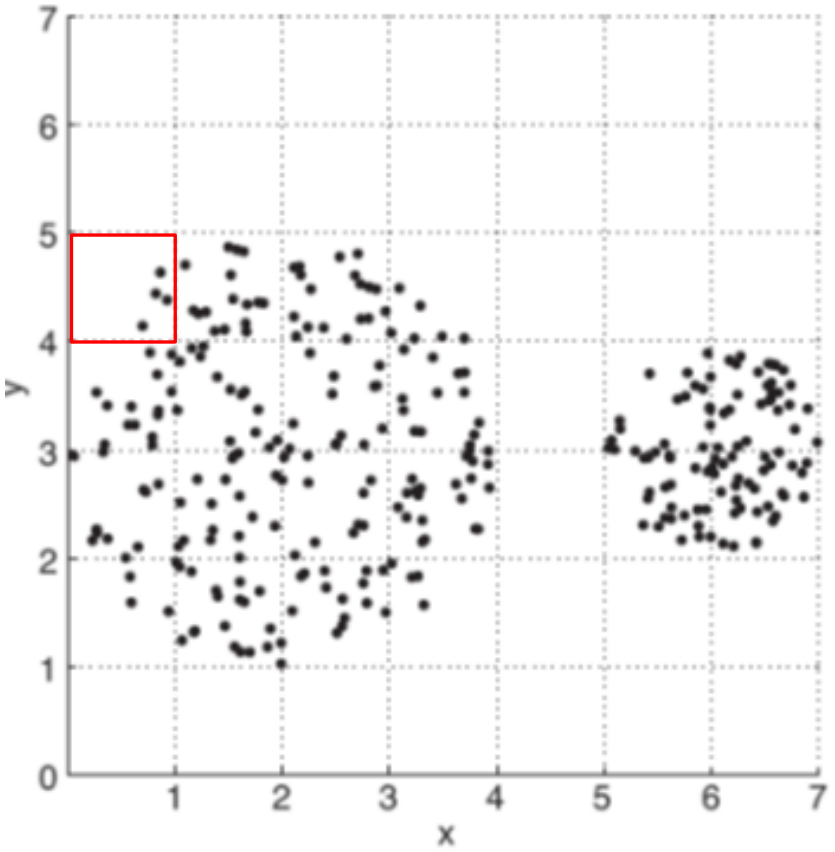
Strengths and Limitations

- Efficient:
  - One pass needed for assignment and counts
  - Need to store only non-empty cells
- Expensive in high dimensional data
- Depends on choice of threshold T
  - If T is too low: cells that should be separated will be joined
  - If T is too high: clusters will be lost
- Depends on cell size and splits: subdivision of attributes into intervals

Strengths and Limitations

- Rectangular grids do not accurately capture spherical shapes

- Make the grid finer but this may show more fluctuation since points inside the cells may not be evenly distributed



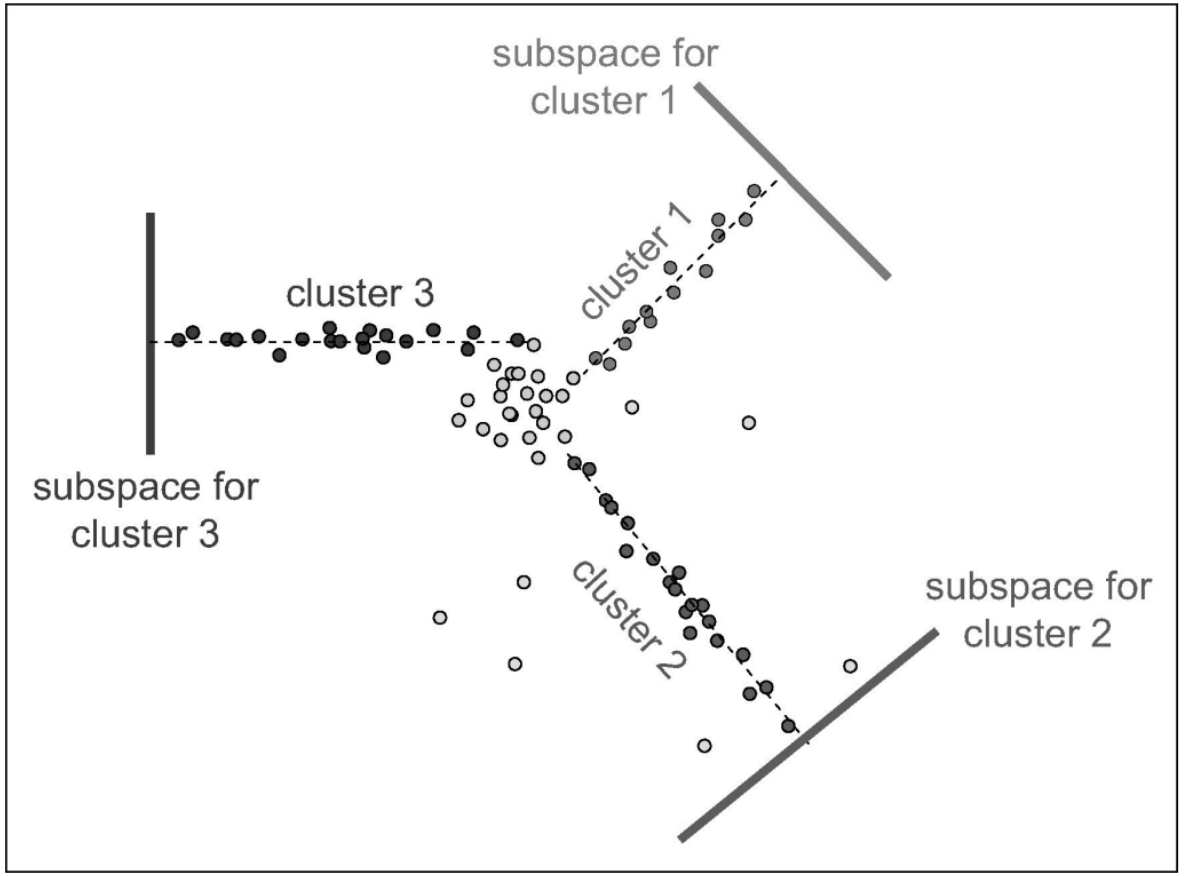
Techniques for Grid-Based Clustering

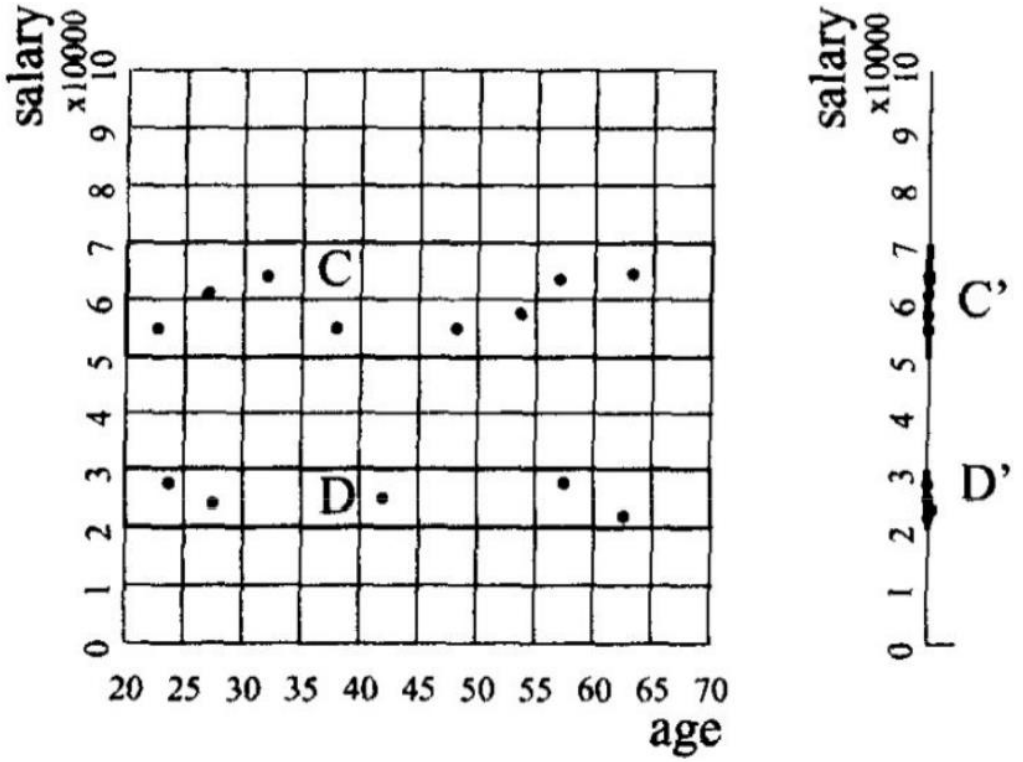
CLIQUE (CLustering in QUEst): It is a density-based and grid-based subspace clustering algorithm:

- Grid-based: It discretizes the data space through a grid
- Density-based: A cluster is a maximal set of connected dense units in a subspace
- Subspace clustering: A subspace cluster is a set of neighboring dense cells in a subspace.
- It automatically find subspaces of a high dimensional data space that allow better clustering than the original space using the apriori principle.

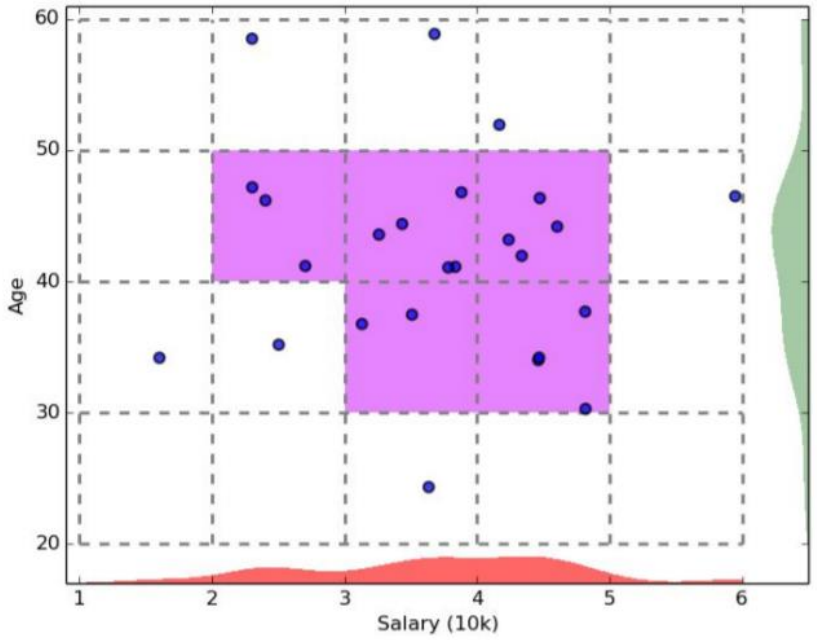
•R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD'98

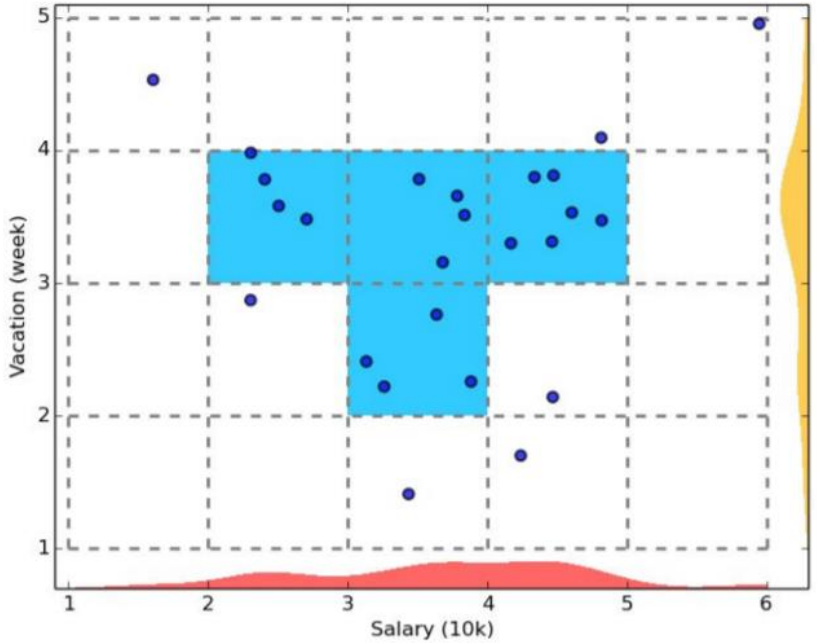
Data Cases - Subspace

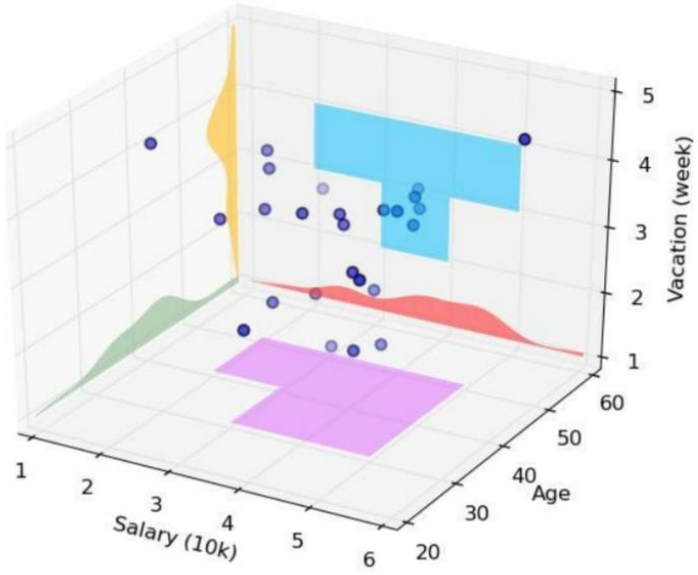




Example of CLIQUE

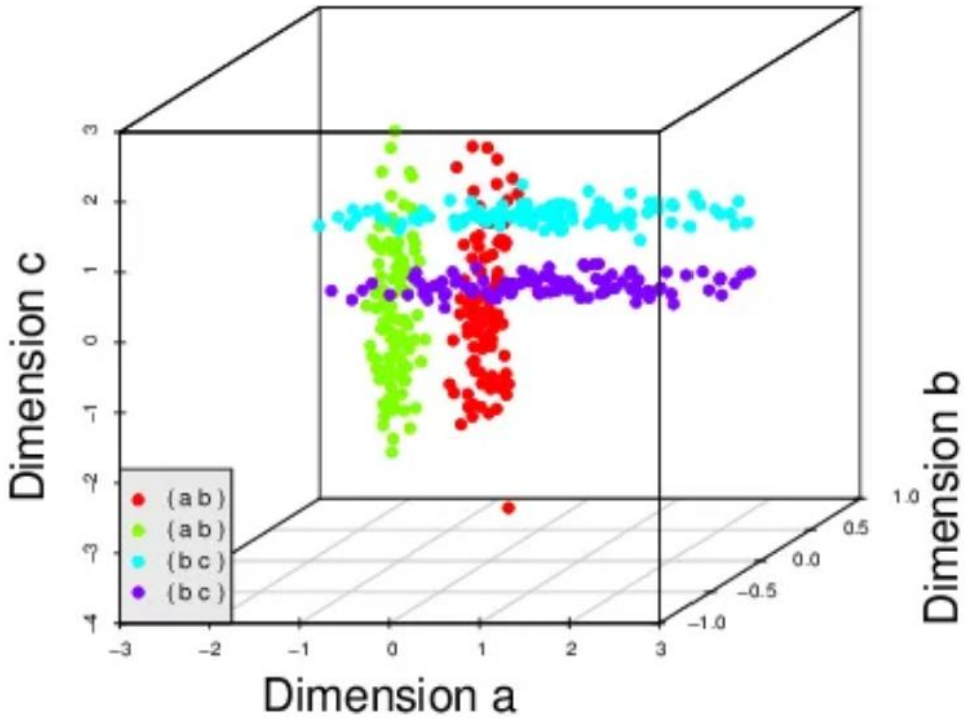


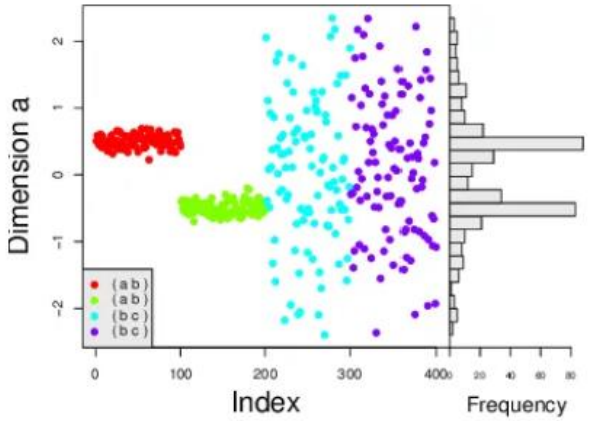




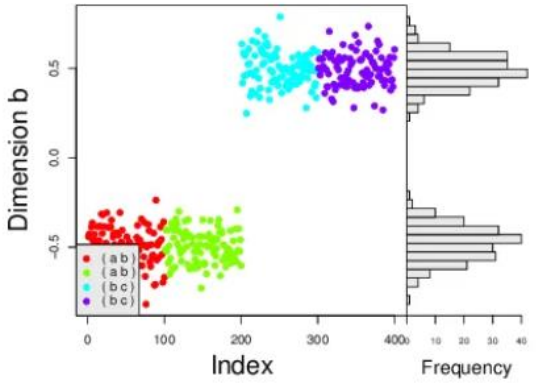
- Apriori principle: If a collection of points S is a cluster in a k-dimensional space, then S is also part of a cluster in any (k-1) dimensional projections of this space

Example

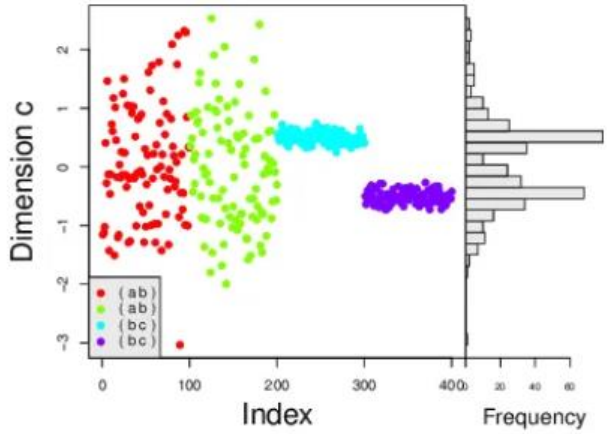




(a) Dimension a

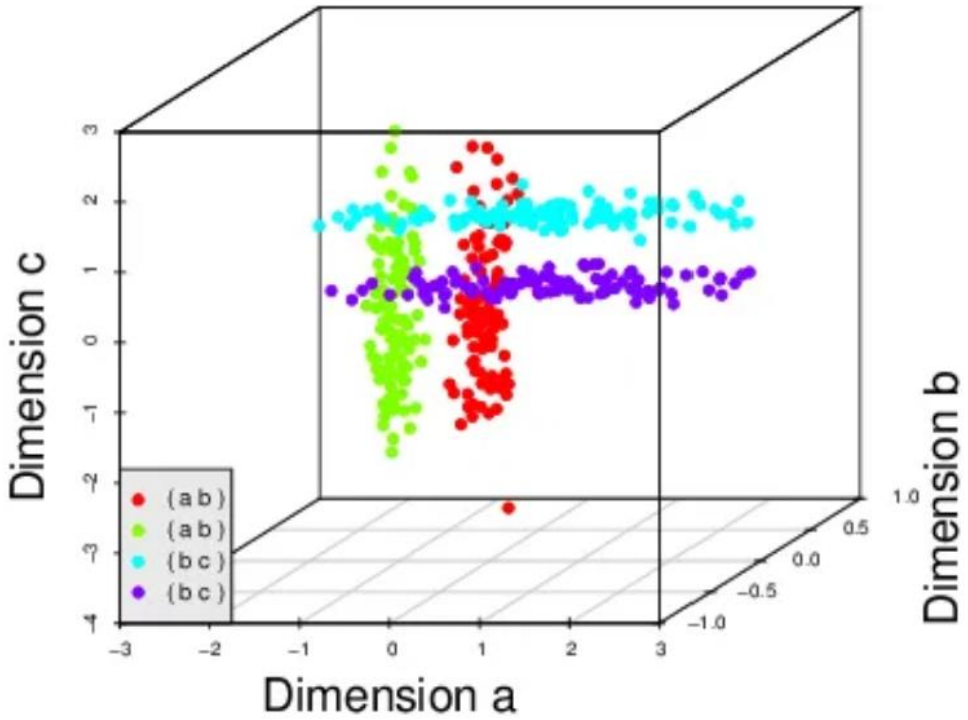


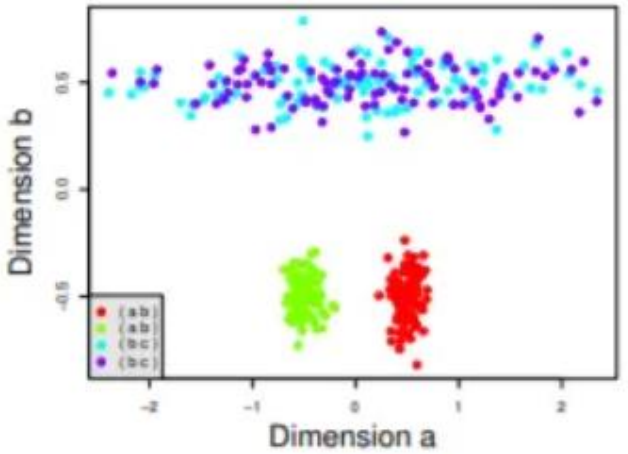
(b) Dimension b



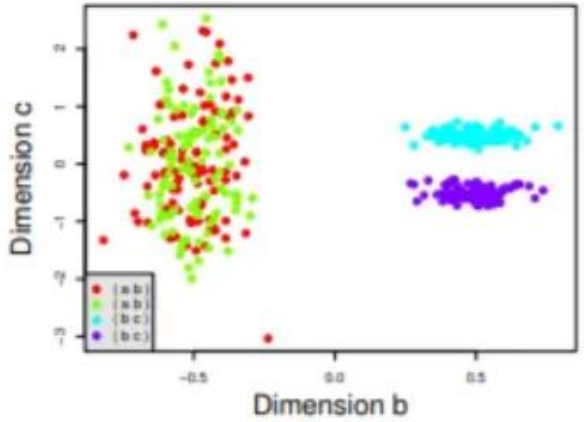
(c) Dimension c

Example

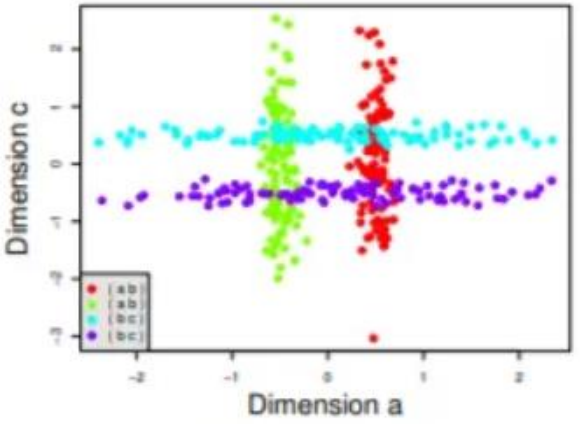




(a) Dims a & b



(b) Dims b & c



(c) Dims a & c

Implementation: **pyclustering.cluster.clique**

Install pyclustering library

conda install -c conda-forge pyclustering

Documentation: https://pypi.org/project/pyclustering/

Exercise: Parameter selection for Digits

- Load Data
- For given k (min_samples)
  - Compute the k-dist for all points
  - Sort them in increasing order
  - Plot the sorted values
    - A sharp change at the value of k-dist that corresponds to a suitable Eps
  - Select this distance as Eps
  - Select k as MinPts

```
from sklearn.datasets import load_digits

from sklearn.manifold import TSNE


digits = load_digits()

digits.data.shape

tsne = TSNE(n_components=2)

xt = tsne.fit_transform(digits.data)
```