UNIVERSITAT DE BARCELONA

Course. Introduction to Machine Learning
Work 1. Clustering Exercise

Session 1

Course 2020-2021

Dr. Maria Salamó Llorente

Dept. Mathematics and Informatics,

Faculty of Mathematics and Informatics,

University of Barcelona

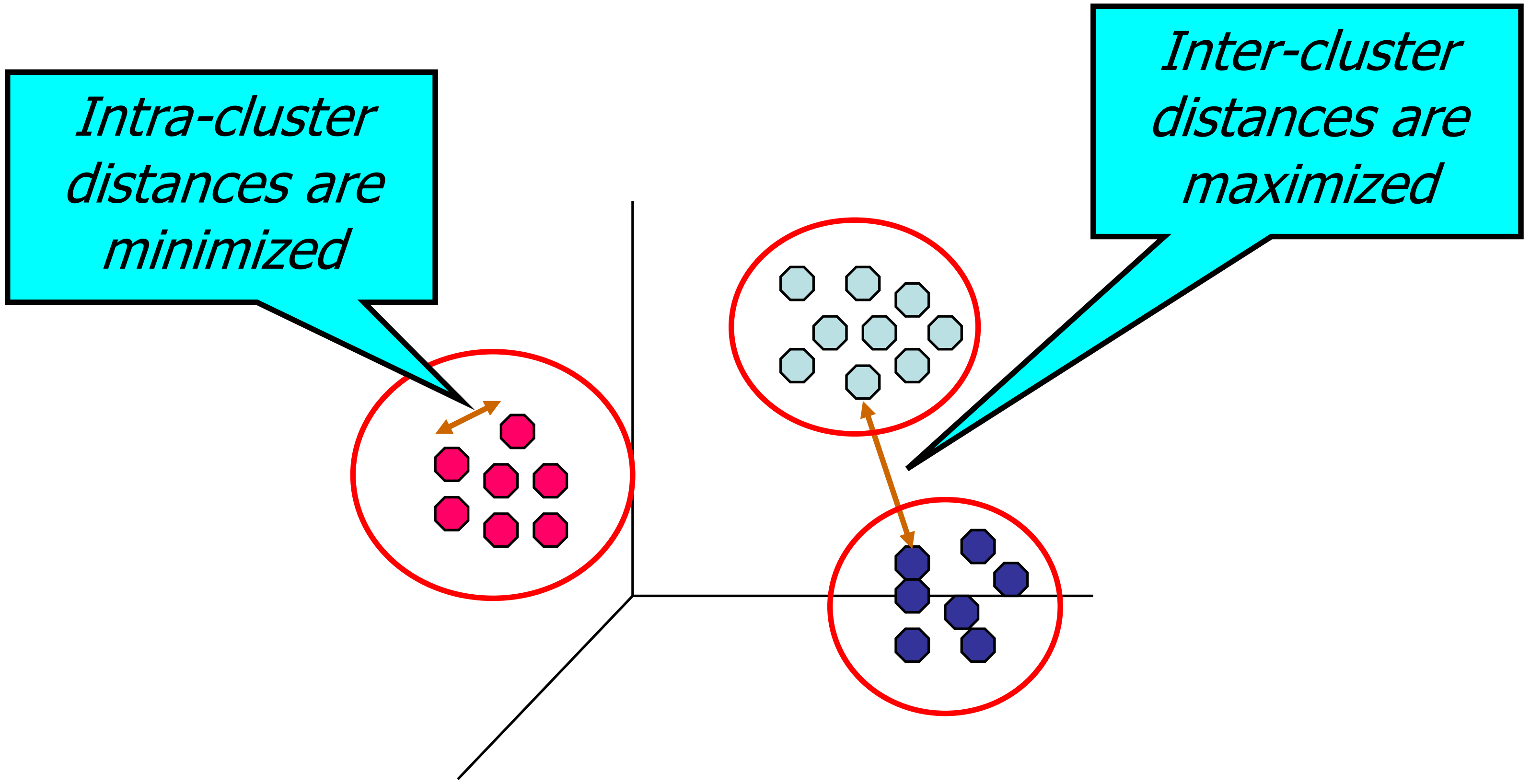
UNIVERSITAT DE BARCELONA

Contents

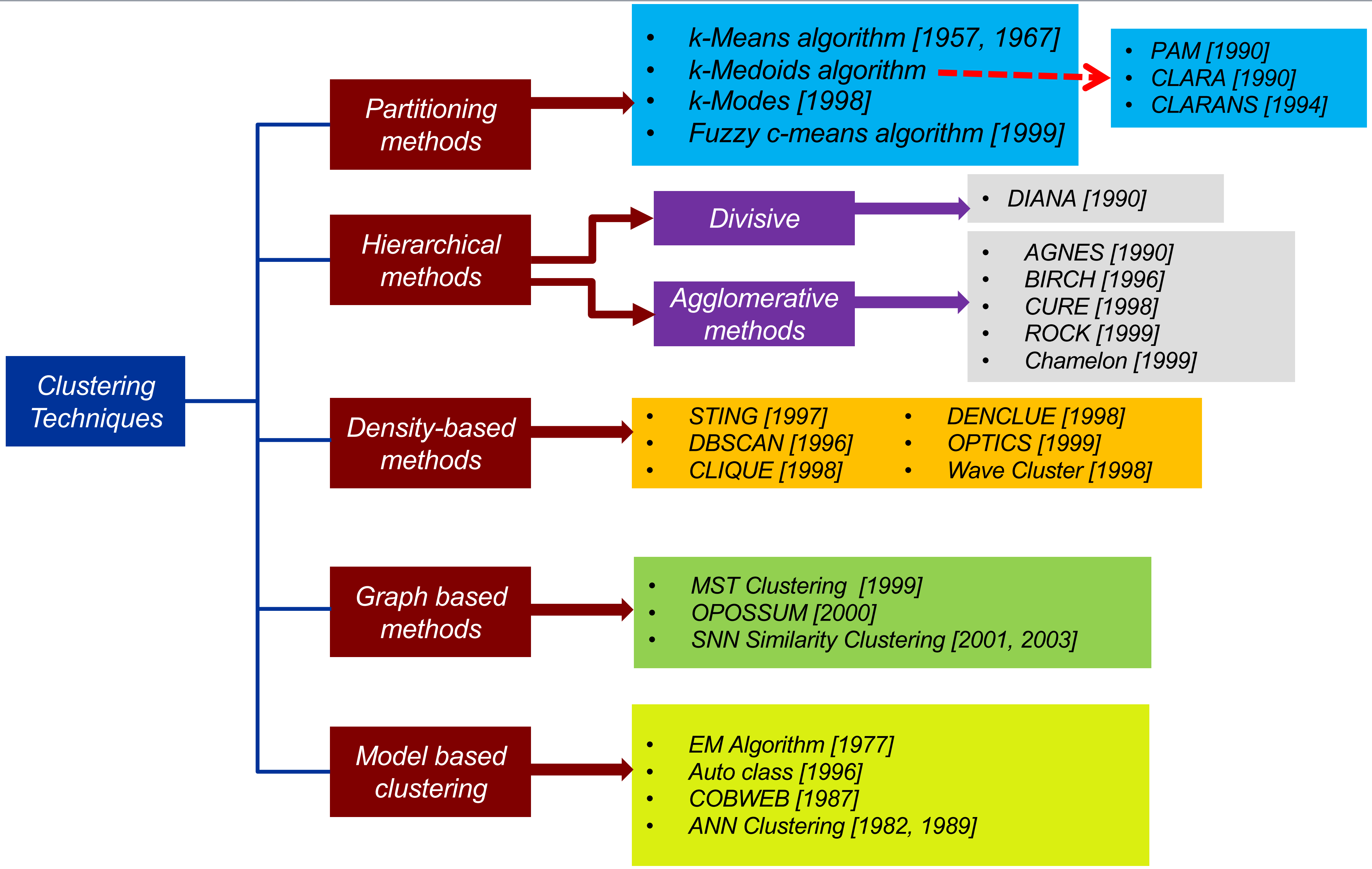
Introduction

What is a Clustering?

- In general a **grouping** of objects such that the objects in a group (cluster) are similar (or related) to one another and different from (or unrelated to) the objects in other groups


Intra-cluster distances are minimized
Inter-cluster distances are maximized

Taxonomy of Clustering Algorithms



Code and Packages

- You need to implement the code using Python 3.6 and Pycharm IDE

- Packages allowed in this exercise:
  - arff_loader
  - numpy
  - pandas
  - scipy
  - sklearn (only for some parts)
  - matplotlib
  - seaborn

Preprocess the data

Data in

- You need to read the .arff file
  - You can implement your own code or use scipy.io.arff.loadarff
- Data needs pre-processing
  - Features may contain **different ranges**
    - Normalize or Standarize the machine learning data
  - Features may have **different types**
    - Categorical, Numerical, and mix-type data
  - Data may contain **missing values**
    - Use the median (for example)

UNIVERSITAT DE BARCELONA

Data pre-processing

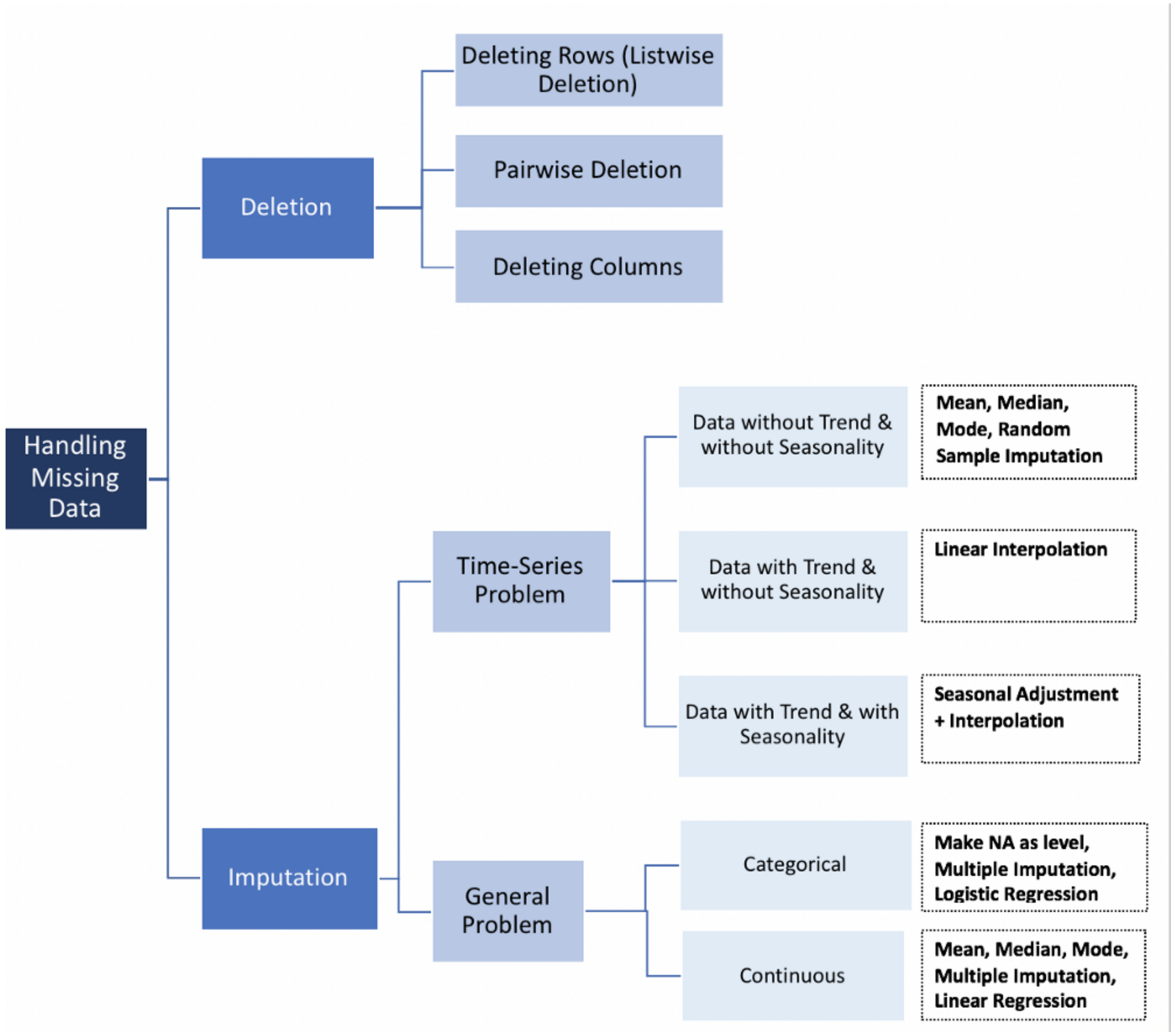
- **To deal with different ranges**
  - Normalize or scale features
- Alternatives
  - **Standardisation**: Standardisation replaces the values by their Z scores. `sklearn.preprocessing.scale`
  - **Mean normalisation**: This distribution will have values between **-1 and 1**with **μ=0**. `sklearn.preprocessing.StandardScaler`
  - **Min-Max scaling**: This scaling brings the value between 0 and 1. `sklearn.preprocessing.MinMaxScaler`
  - **Unit vector**: Scaling is done considering the whole feature vector to be of unit length. `sklearn.preprocessing.Normalizer`

Data pre-processing

- **To deal with different types**
- Alternatives
  - **Label encoding**: convert to a number
    `sklearn.preprocessing.LabelEncoder`
  - **One hot encoding**: where a categorical variable is converted into a binary vector, each possible value of the categorical variable becomes the variable itself with default value of zero and the variable which was the value of the categorical variable will have the value 1.
    `sklearn.preprocessing.OneHotEncoder`

Data pre-processing

- **To deal with missing values**


Handling Missing Data
Deletion
Deleting Rows (Listwise Deletion)
Pairwise Deletion
Deleting Columns
Imputation
Time-Series Problem
Data without Trend & without Seasonality
Mean, Median, Mode, Random Sample Imputation
Data with Trend & without Seasonality
Linear Interpolation
Data with Trend & with Seasonality
Seasonal Adjustment + Interpolation
General Problem
Categorical
Make NA as level, Multiple Imputation, Logistic Regression
Continuous
Mean, Median, Mode, Multiple Imputation, Linear Regression

DBSCAN

Density-Based Clustering

Using sklearn

DBSCAN

Some Links

- http://www2.cs.uh.edu/~ceick/7363/Papers/dbscan.pdf
- https://youtu.be/sKRUfsc8zp4
- https://youtu.be/6jl9KkmgDIw
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN

UNIVERSITAT DE BARCELONA

Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as **density-connected points** or based on an explicitly constructed density function
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - DENCLUE: Hinneburg & D. Keim (KDD'98/2006)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - CLIQUE: Agrawal, et al. (SIGMOD'98)

DBSCAN: Density-Based Clustering

- DBSCAN is a Density-Based Clustering algorithm

- Reminder: In density based clustering we partition points into dense regions separated by not-so-dense regions.

- Important Questions:
  - How do we measure density?
  - What is a dense region?

- DBSCAN:
  - Density at point p: number of points within a circle of radius Eps
  - Dense Region: A circle of radius Eps that contains at least MinPts points

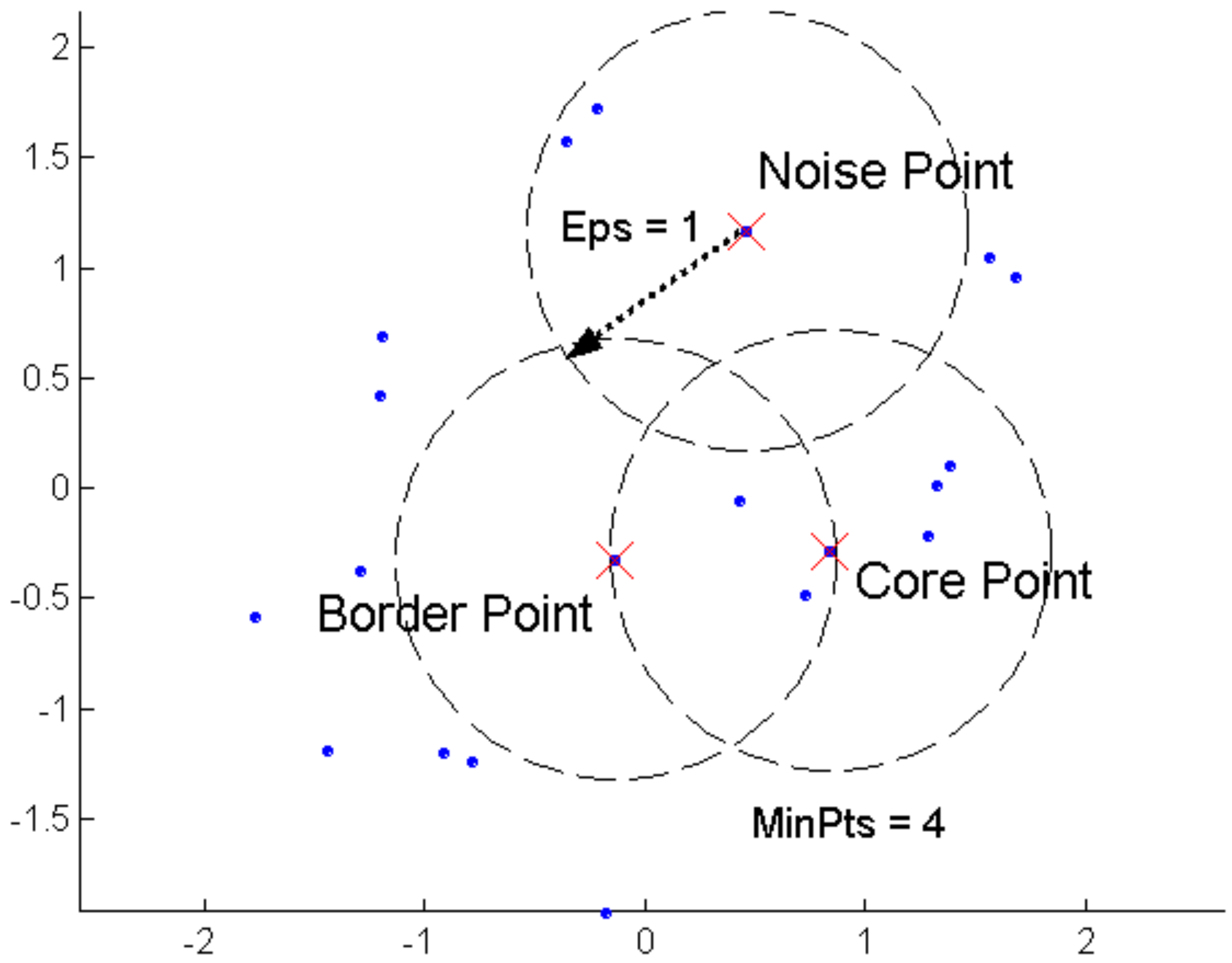
UNIVERSITAT DE BARCELONA

DBSCAN: Preliminary concepts

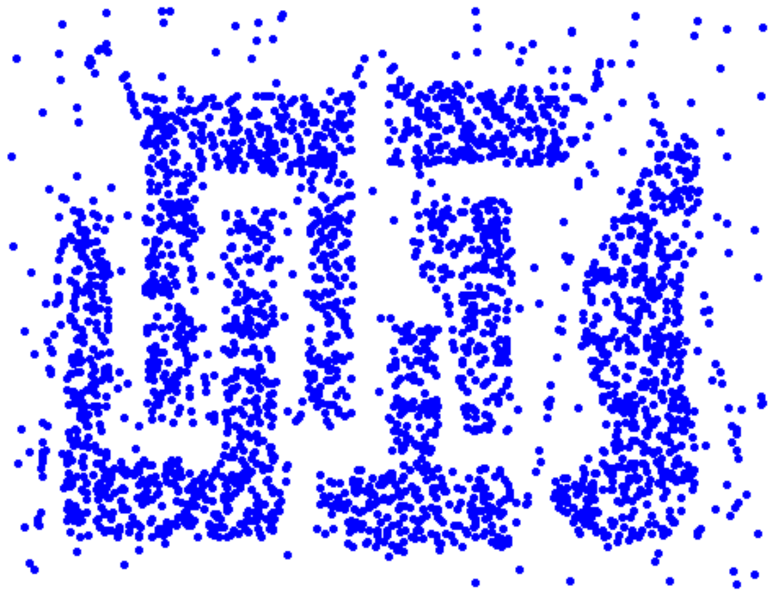
Characterization of points

- Density = number of points within a specified radius (Eps)
- A point is a **core point** if it has more than a specified number of points (*MinPts*) within Eps
  - These points belong in a dense region and are at the interior of a cluster
- A **border point** has fewer than *MinPts* within Eps, but is in the neighborhood of a core point
- A **noise point** is any point that is not a core point or a border point

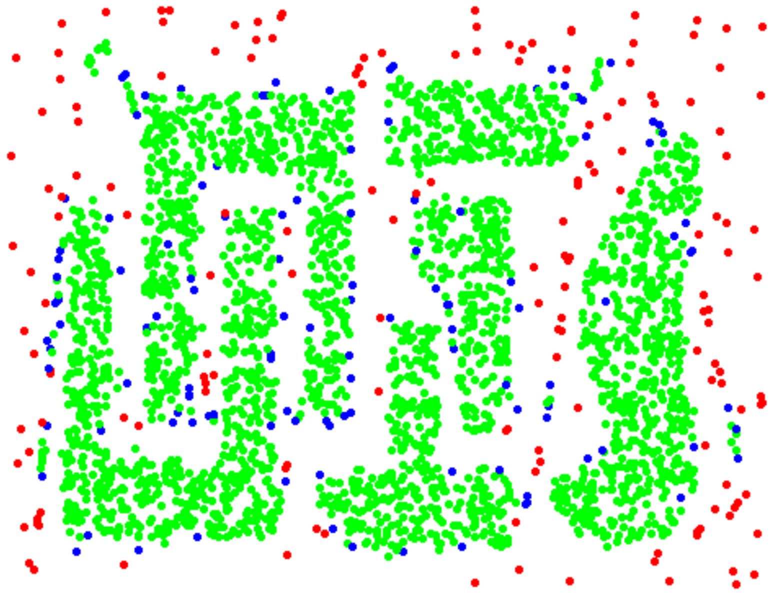
DBSCAN: Core, Border, and Noise Points



DBSCAN: Core, Border and Noise Points



Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

Parameter Estimation

- Parameters must be specified by the user
  - Ɛ = physical distance (radius),
  - *minPts* = desired minimum cluster size

minPts

- derived from the number of dimensions *D* in the data set, as *minPts* ≥ *D* + *1*
- *minPts* = *1* does not make sense, as then every point on its own will already be a cluster
- *minPts* must be chosen at least 3. Larger is better.
- larger the dataset, the larger the value of *minPts* should be chosen

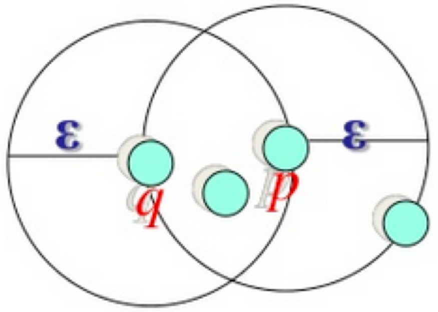
Ɛ

- value can be chosen by using a k-distance graph
- If Ɛ is chosen much too small, a large part of the data will not be clustered
- If too high value, majority of objects will be in the same cluster
- In general, small values of Ɛ are preferable

Concepts: Ɛ-Neighborhood

- Ɛ-Neighborhood: Objects within a radius of Ɛ from an object (epsilon-neighborhood)

- Core objects: Ɛ-Neighborhood of an object contains at least *MinPts* of objects



ε-Neighborhood of *p*
ε-Neighborhood of *q*

p is a core object (MinPts = 4)

q is not a core object

DBSCAN Algorithm (simplified view for teaching)

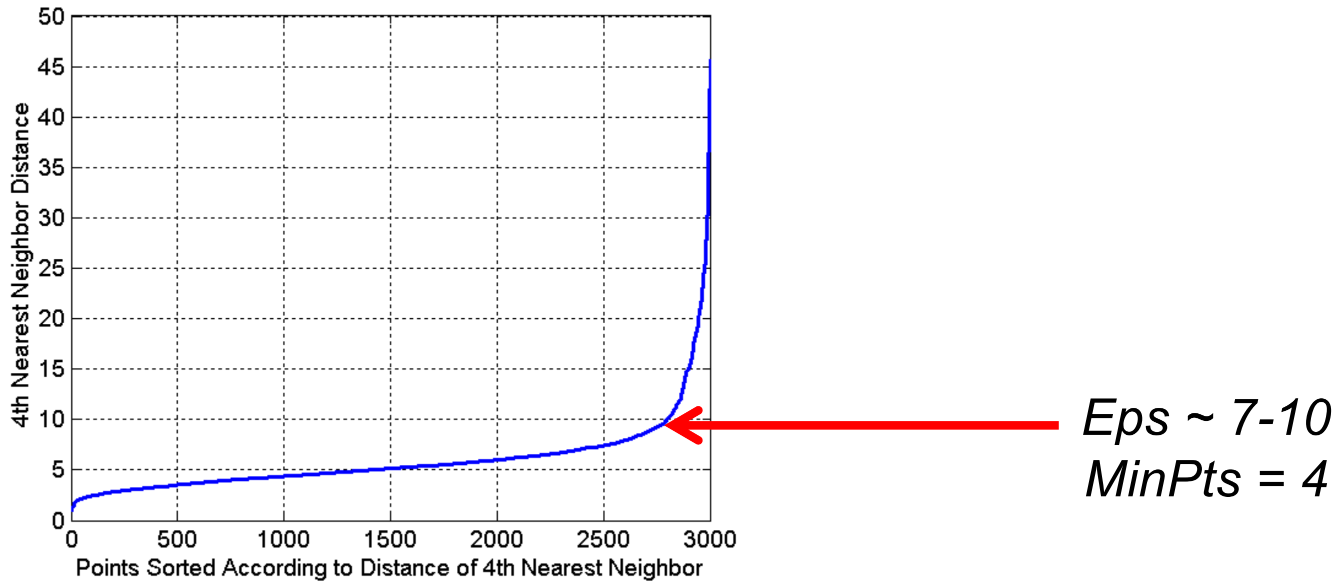
1. Create a graph whose nodes are the points to be clustered
2. For each core-point c create an edge from c to every point p in the **ε-neighborhood** of c
3. Set N to the nodes of the graph;
4. If N does not contain any core points terminate
5. Pick a core point c in N
6. Let X be the set of nodes that can be reached from c by going forward;
    1. create a cluster containing X∪{c}
    2. N=N/(X∪{c})
7. Continue with step 4

Remark: points that are not assigned to any cluster are outliers;

UNIVERSITAT DE BARCELONA

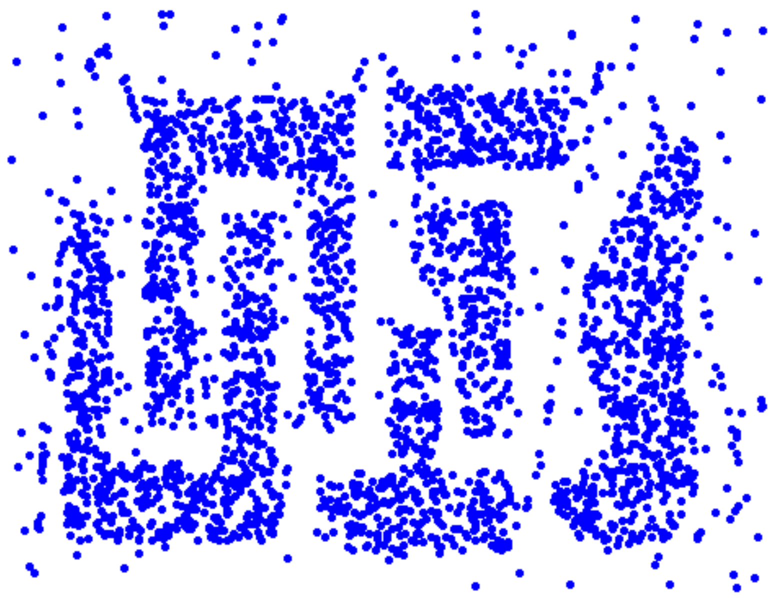
DBSCAN: Determining EPS and *MinPts*

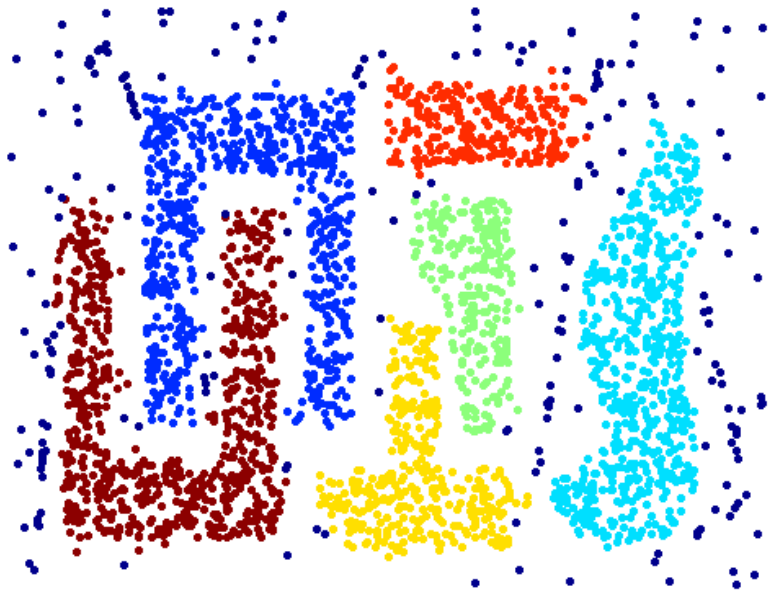
- Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor
- Find the distance d where there is a “knee” in the curve
  - Eps = d, MinPts = k



UNIVERSITAT DE BARCELONA

When DBSCAN Works Well

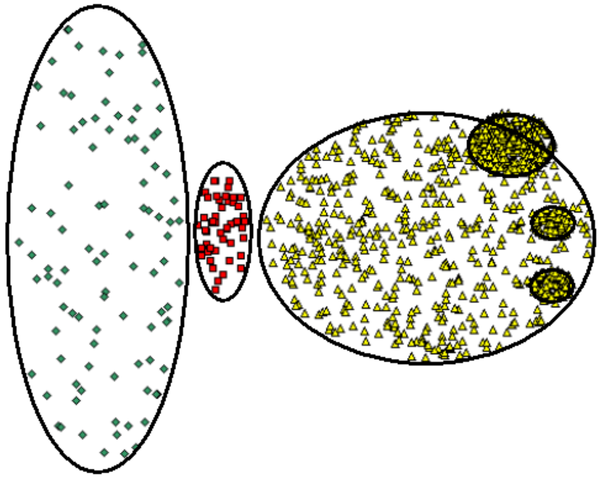

Original Points


Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

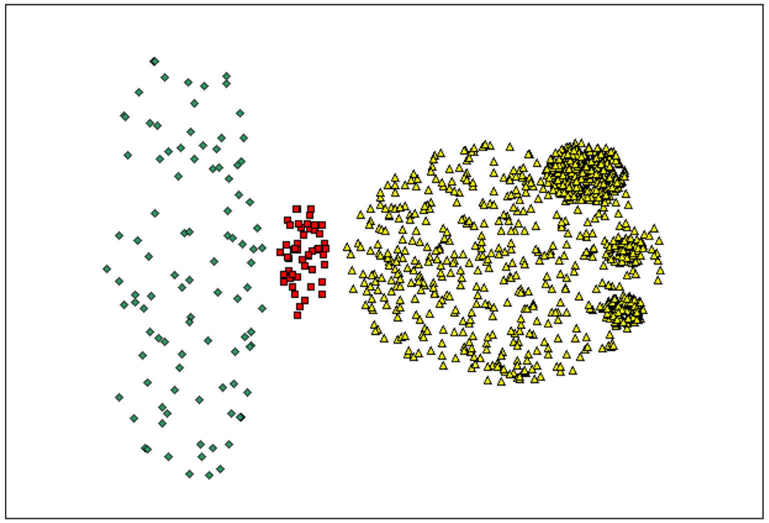
UNIVERSITAT DE BARCELONA

When DBSCAN Does NOT Work Well

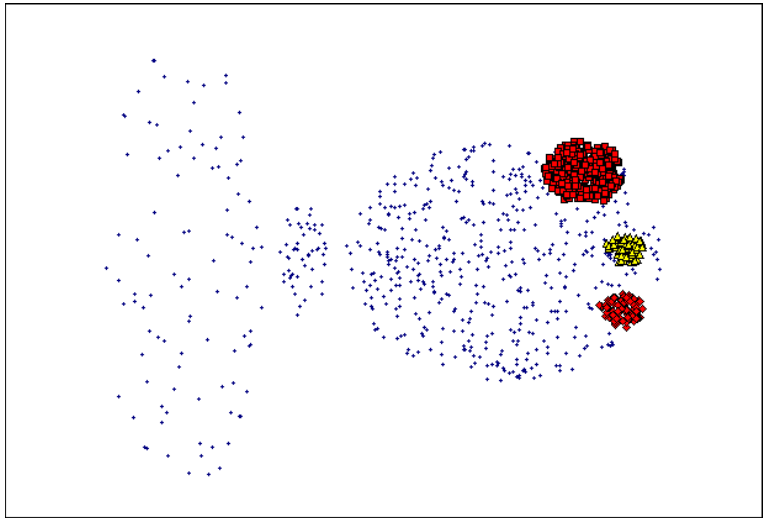


Original Points

- Varying densities
- High-dimensional data



(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

UNIVERSITAT DE BARCELONA

Complexity DBSCAN

- Time Complexity: `O(n²)`
  - For each point it has to be determined if it is a core point
  - Can be reduced to `O(n*log(n))` in lower dimensional spaces by using efficient data structures (where n is the number of objects to be clustered);

- Space Complexity: `O(n)`