DATABASE SYSTEMS GROUP

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

LMU

Knowledge Discovery in Databases

SS 2016

Chapter 4: Clustering

Lecture: Prof. Dr. Thomas Seidl

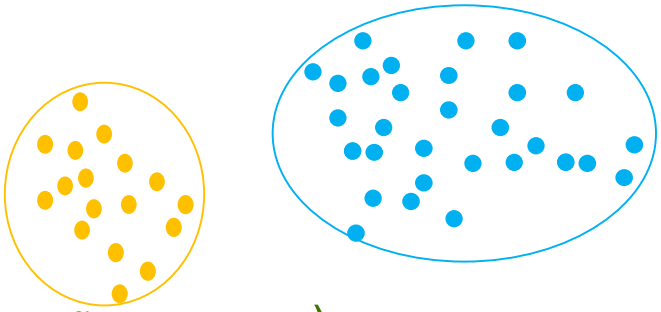
Tutorials: Julian Busch, Evgeniy Faerman,
Florian Richter, Klaus Schmid

Contents

What is Clustering?

Grouping a set of data objects into clusters

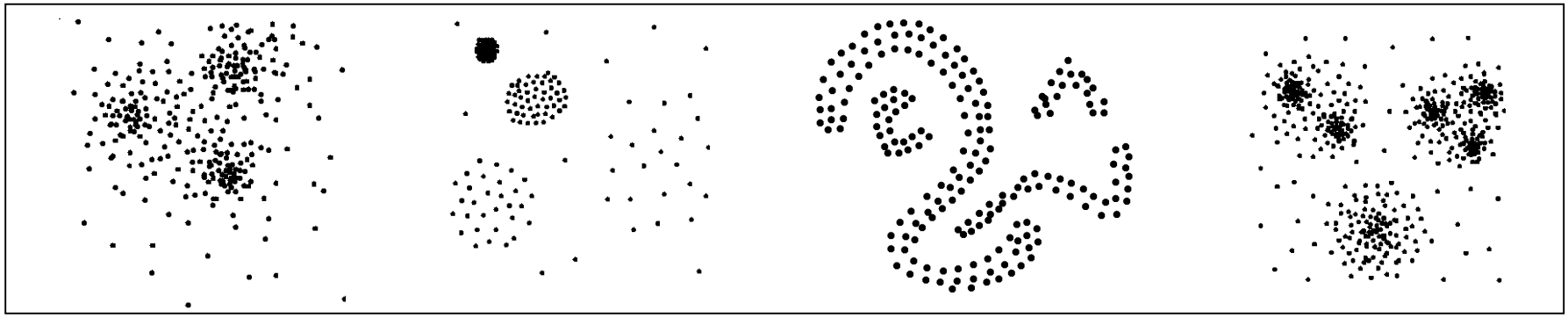
- Cluster: a collection of data objects
  1) *Similar* to one another within the same cluster
  2) *Dissimilar* to the objects in other clusters



Clustering = ***unsupervised "classification"*** (no predefined classes)

Typical usage

- As a *stand-alone tool* to get insight into data distribution
- As a *preprocessing step* for other algorithms



General Applications of Clustering

Preprocessing – as a data reduction (instead of sampling)

- Image data bases (color histograms for filter distances)
- Stream clustering (handle endless data sets for offline clustering)

Pattern Recognition and Image Processing

Spatial Data Analysis

- create thematic maps in Geographic Information Systems by clustering feature spaces
- detect spatial clusters and explain them in spatial data mining

Business Intelligence (especially market research)

WWW

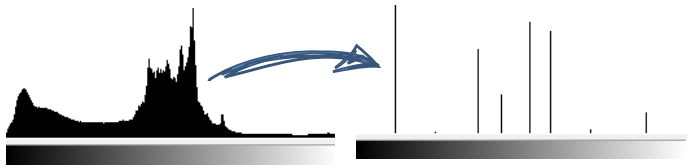
- Documents (Web Content Mining)
- Web-logs (Web Usage Mining)

Biology

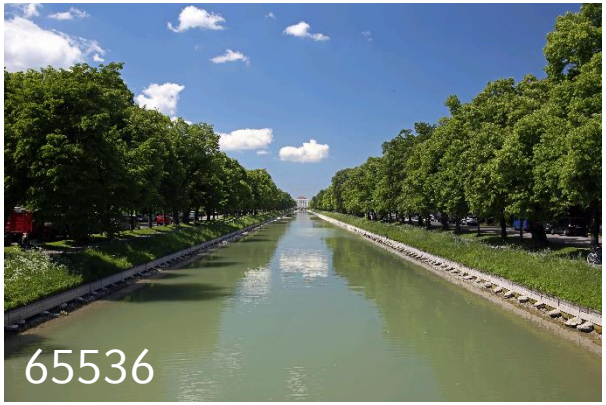
- Clustering of gene expression data

An Application Example: Downsampling Images

- Reassign color values to k distinct colors
- Cluster pixels using color difference, not spatial data



58483 KB



19496 KB

9748 KB

Major Clustering Approaches

Partitioning algorithms
- Find k partitions, minimizing some objective function

Probabilistic Model-Based Clustering (EM)

Density-based
- Find clusters based on connectivity and density functions

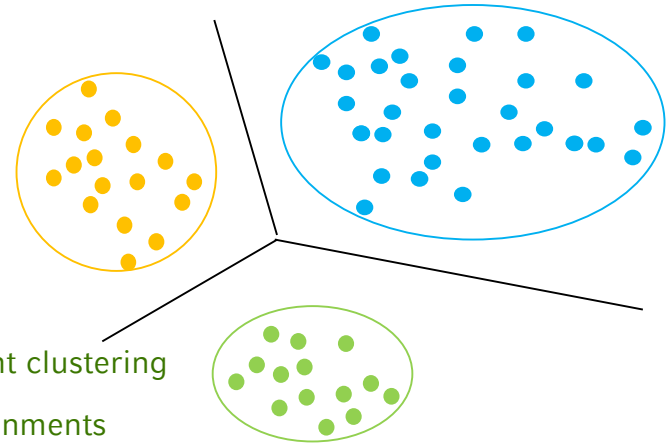
Hierarchical algorithms
- Create a hierarchical decomposition of the set of objects

Other methods
- Grid-based
- Neural networks (SOM's)
- Graph-theoretical methods
- Subspace Clustering
- . . .

Contents

Partitioning Algorithms: Basic Concept

- Goal: Construct a partition of a database D of n objects into a set of k ($k < n$) clusters $C_1, \ldots, C_k$ ($C_i \subset D, C_i \cap C_j = \emptyset \Leftrightarrow C_i \neq C_j, \bigcup C_i = D$) minimizing an objective function.
  - Exhaustively enumerating all possible partitions into k sets in order to find the global minimum is too expensive.



- Popular heuristic methods:
  - Choose k representatives for clusters, e.g., randomly
  - Improve these initial representatives iteratively:
    - Assign each object to the cluster it "fits best" in the current clustering
    - Compute new cluster representatives based on these assignments
    - Repeat until the change in the objective function from one iteration to the next drops below a threshold

- Examples of representatives for clusters
  - k-means: Each cluster is represented by the center of the cluster
  - k-medoid: Each cluster is represented by one of its objects

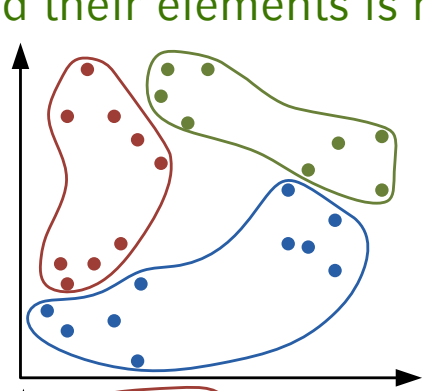
Contents

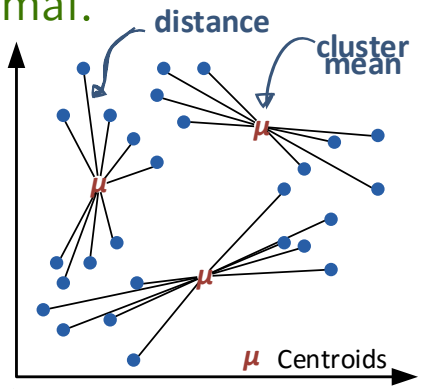
K-Means Clustering: Basic Idea

Idea of K-means: find a clustering such that the *within-cluster variation* of each cluster is small and use the *centroid* of a cluster as representative.

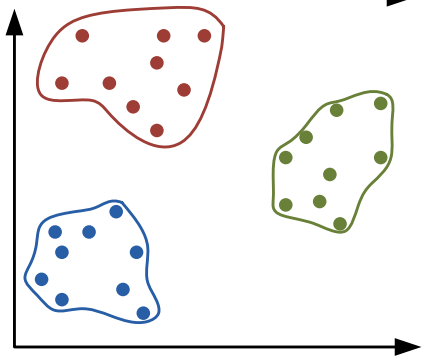
Objective: For a given *k*, form *k* groups so that the sum of the (squared) distances between the mean of the groups and their elements is minimal.

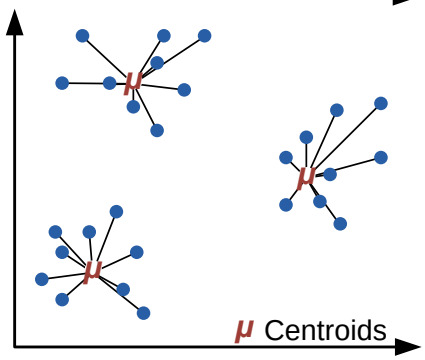
Poor Clustering
(large sum of distances)





Optimal Clustering
(minimal sum of distances)





S.P. Lloyd: Least squares quantization in PCM. In IEEE Information Theory, 1982 (original version: technical report, Bell Labs, 1957)
J. MacQueen: *Some methods for classification and analysis of multivariate observation*, In Proc. of the 5th Berkeley Symp. on Math. Statist. and Prob., 1967.

K-Means Clustering: Basic Notions

Objects $p = (p_1, \ldots, p_d)$ are points in a d-dimensional vector space

(the mean μ_S of a set of points S must be defined: $\mu_S = \frac{1}{|S|}\sum_{p \in S} p$)

Measure for the compactness of a **cluster** C_j (sum of squared errors):

$$SSE(C_j) = \sum_{p \in C_j} dist\left(p, \mu_{C_j}\right)^2$$

Measure for the compactness of a **clustering** $\mathcal{C}$:

$$SSE(\mathcal{C}) = \sum_{C_j \in \mathcal{C}} SSE(C_j) = \sum_{p \in DB} dist\left(p, \mu_{C(p)}\right)^2$$

Optimal Partitioning: $\underset{\mathcal{C}}{\operatorname{argmin}}\, SSE(\mathcal{C})$

Optimizing the within-cluster variation is computationally challenging (NP-hard) → use efficient heuristic algorithms

K-Means Clustering: Algorithm

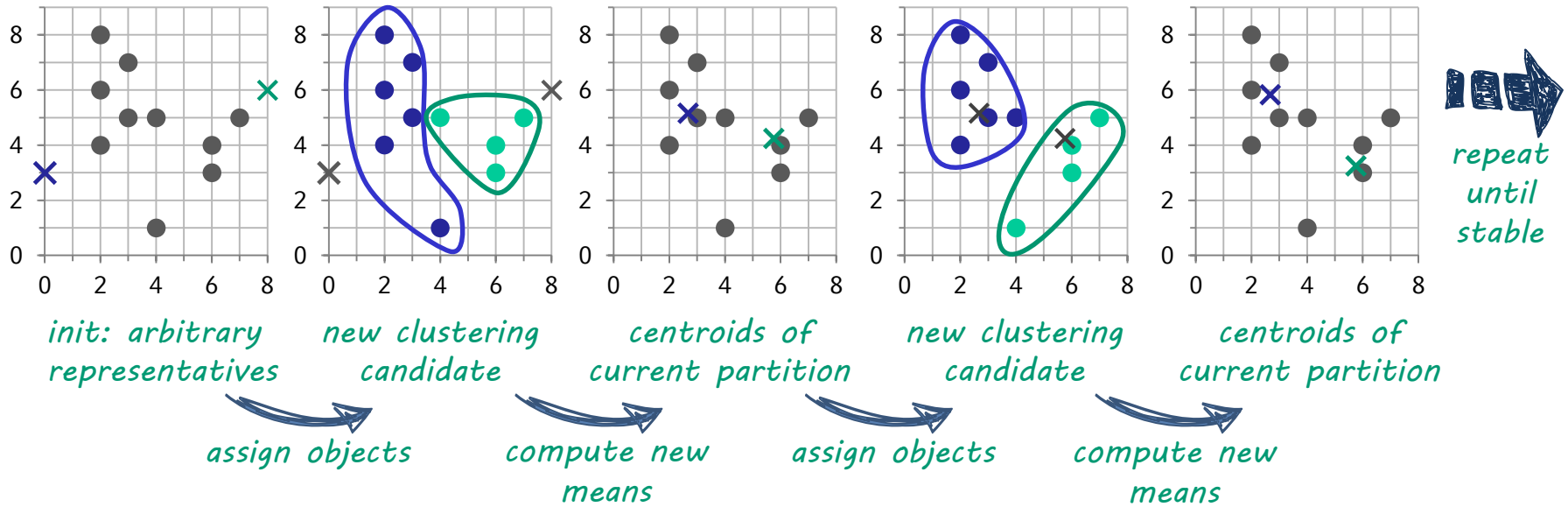
k-Means algorithm (Lloyd's algorithm):

Given *k*, the *k*-means algorithm is implemented in 2 main steps:

Initialization: Choose *k* arbitrary representatives

Repeat until representatives do not change:

1. Assign each object to the cluster with the nearest representative.
2. Compute the centroids of the clusters of the current partitioning.



K-Means Clustering: Discussion

Strengths

- Relatively efficient: $O(tkn)$, where n = # objects, k = # clusters, and t = # iterations
- Typically: k, $t \ll n$
- Easy implementation

Weaknesses

- Applicable only when mean is defined
- Need to specify k, the number of clusters, in advance
- Sensitive to noisy data and outliers
- Clusters are forced to convex space partitions (Voronoi Cells)
- Result and runtime strongly depend on the initial partition; often terminates at a *local optimum* – however: methods for a good initialization exist

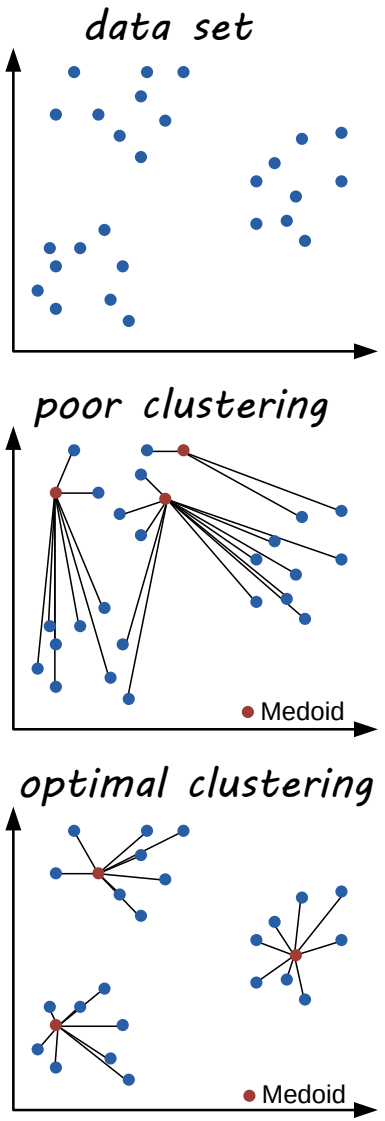
Several variants of the *k*-means method exist, e.g., ISODATA

- Extends *k*-means by methods to eliminate very small clusters, merging and split of clusters; user has to specify additional parameters

Contents

K-Medoid, *K*-Modes, *K*-Median Clustering: Basic Idea

- Problems with K-Means:
  - Applicable only when mean is defined (vector space)
  - Outliers have a strong influence on the result
- The influence of outliers is intensified by the use of the *squared* error → use the absolute error (total distance instead): $TD(C) = \sum_{p \in C} dist(p, m_{c(p)})$ and $TD(\mathcal{C}) = \sum_{C_i \in \mathcal{C}} TD(C_i)$
- Three alternatives for using the Mean as representative:
  - *Medoid*: representative object "in the middle"
  - *Mode*: value that appears most often
  - *Median*: (artificial) representative object "in the middle"
- Objective as for k-Means: Find *k* representatives so that, the sum of the distances between objects and their closest representative is minimal.



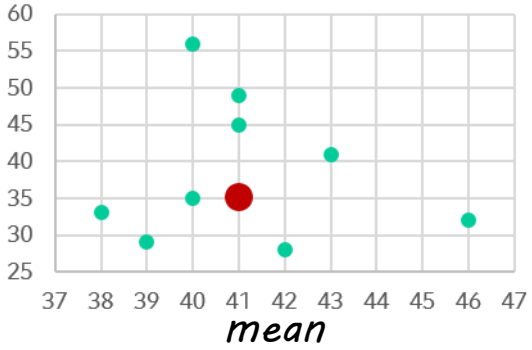
K-Median Clustering

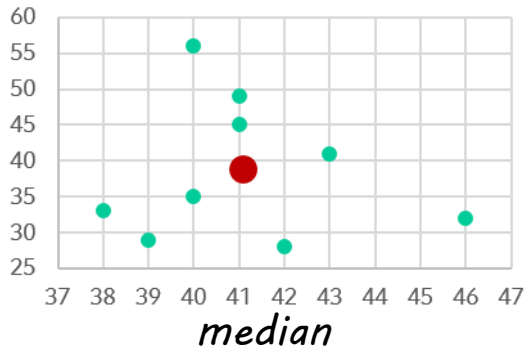
Problem: Sometimes, data is not numerical

Idea: If there is an ordering on the data $X = \{x_1, x_2, x_3, \ldots, x_n\}$, use median instead of mean

$$Median(\{x\}) = x$$
$$Median(\{x, y\}) \in \{x, y\}$$
$$Median(X) = Median(X - \min X - \max X), \qquad if\ |X| > 2$$

- A median is computed in each dimension independently and can thus be a combination of multiple instances
  → median can be efficiently computed for ordered data
- Different strategies to determine the "middle" in an array of even length possible





K-Mode Clustering: First Approach [Huang 1997]

Given: $X \subseteq \Omega = A_1 \times A_2 \times \cdots \times A_d$ is a set of n objects, each described by d categorical attributes A_i $(1 \leq i \leq d)$

Mode: a mode of X is a vector $M = [m_1, \mathrm{m}_2, \cdots, m_d] \in \Omega$ that minimizes

$$d(M, X) = \sum_{x_i \in X} d(x_i, M)$$

where d is a distance function for categorical values (e.g. Hamming Dist.)

→ Note: M is not necessarily an element of X

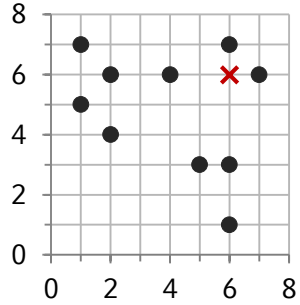
Theorem to determine a Mode: let $f(c, j, X) = \frac{1}{n} \cdot |\{x \in X | \, x[j] = c\}|$ be the relative frequency of category c of attribute A_j in the data, then:

$$d(M, X) \text{ is minimal} \Leftrightarrow \forall j \in \{1, \ldots, d\}: \forall c \in A_j: f(\mathrm{m_j}, \mathrm{j}, \mathrm{X}) \geq f(c, j, X)$$

→ this allows to use the k-means paradigm to cluster categorical data without loosing its efficiency

→ Note: the mode of a dataset might be not unique

K-Modes algorithm proceeds similar to k-Means algorithm



Huang, Z.: *A Fast Clustering Algorithm to Cluster very Large Categorical Data Sets in Data Mining*, In DMKD, 1997.

K-Mode Clustering: Example

Employee-ID	Profession	Household Pets
#133	Technician	Cat
#134	Manager	None
#135	Cook	Cat
#136	Programmer	Dog
#137	Programmer	None
#138	Technician	Cat
#139	Programmer	Snake
#140	Cook	Cat
#141	Advisor	Dog

Profession: (**Programmer: 3**, Technician: 2, Cook: 2, Advisor: 1, Manager:1)
Household Pet: (**Cat: 4**, Dog: 2, None: 2, Snake: 1)

Mode is (Programmer, Cat)
Remark: (Programmer, Cat) ∉ DB

K-Medoid Clustering: PAM Algorithm

Partitioning Around Medoids [Kaufman and Rousseeuw, 1990]

- Given *k*, the *k*-medoid algorithm is implemented in 3 steps:
  - Initialization: Select *k* objects arbitrarily as initial medoids (representatives)
  - assign each remaining (non-medoid) object to the cluster with the nearest representative
  - compute $TD_{current}$
- Problem of PAM: high complexity ($O(tk(n-k)^2)$)

Kaufman L., Rousseeuw P. J., *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, 1990.

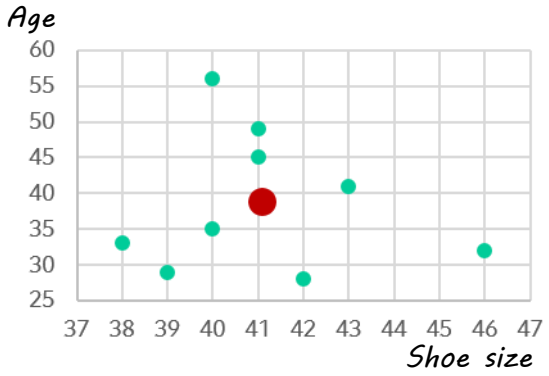
Algorithmus PAM

```
PAM(Punktmenge D, Integer k)
    Initialisiere die k Medoide;
    TD_Änderung := −∞;
    while TD_Änderung < 0 do
        Berechne für jedes Paar (Medoid M, Nicht-Medoid N)
          den Wert TD_{N↔M};
        Wähle das Paar (M, N), für das der Wert
          TD_Änderung := TD_{N↔M} − TD minimal ist;
        if TD_Änderung < 0 then
            ersetze den Medoid M durch den Nicht-Medoid N;
            Speichere die aktuellen Medoide als die bisher beste
              Partitionierung;
    return Medoide;
```

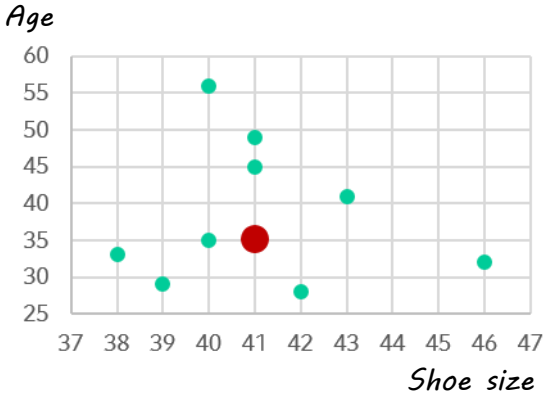
Algorithmus CLARANS

```
CLARANS(Punktmenge D, Integer k,
        Integer numlocal, Integer maxneighbor)
   for r from 1 to numlocal do
      wähle zufällig k Objekte als Medoide; i := 0;
      while i < maxneighbor do
         Wähle zufällig (Medoid M, Nicht-Medoid N);
         Berechne TD_Änderung := TD_{N↔M} – TD;
         if TD_Änderung < 0 then
           ersetze M durch N;
           TD := TD_{N↔M}; i := 0;
         else i:= i + 1;
      if TD < TD_best then
         TD_best := TD; Speichere aktuelle Medoide;
   return Medoide;
```

K-Means/Medoid/Mode/Median overview

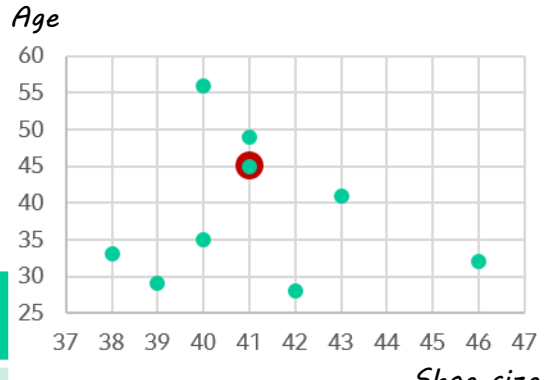


mean



median

Employee-ID	Profession	Shoe size	Age
#133	Technician	42	28
#134	Manager	41	45
#135	Cook	46	32
#136	Programmer	40	35
#137	Programmer	41	49
#138	Technician	43	41
#139	Programmer	39	29
#140	Cook	38	33
#141	Advisor	40	56



medoid

Profession: Programmer
Shoe size: 40/41
Age: n.a.

mode

K-Means/Median/Mode/Medoid Clustering: Discussion

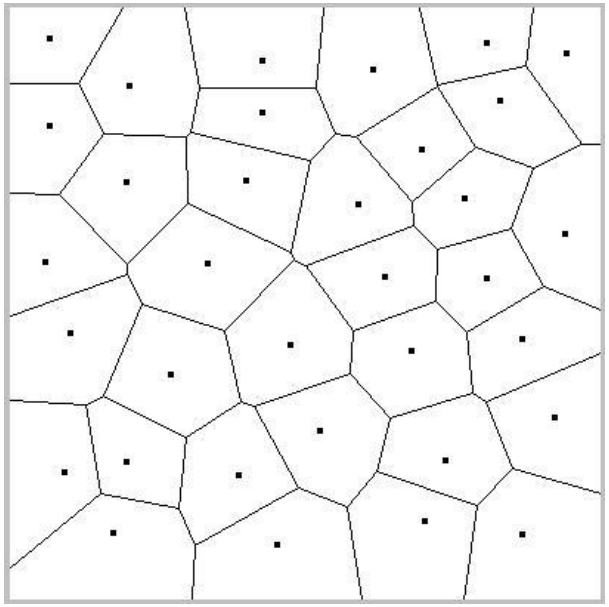
	k-Means	*k*-Median	K-Mode	K-Medoid
data	numerical data (mean)	ordered attribute data	categorical attribute data	metric data
efficiency	high $O(tkn)$	high $O(tkn)$	high $O(tkn)$	low $O(tk(n-k)^2)$
sensitivity to outliers	high	low	low	low

- Strength
  - Easy implementation (→ many variations and optimizations in the literature)
- Weakness
  - Need to specify *k*, the number of clusters, in advance
  - Clusters are forced to convex space partitions (Voronoi Cells)
  - Result and runtime strongly depend on the initial partition; often terminates at a *local optimum* – however: methods for a good initialization exist

Voronoi Model for convex cluster regions

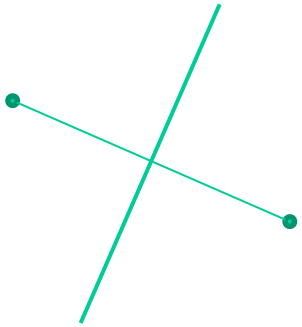
Definition: Voronoi diagram

- For a given set of points $P = \{p_i | \; i = 1 \ldots k\}$ (here: cluster representatives), a Voronoi diagram partitions the data space in Voronoi cells, one cell per point.
- The cell of a point $p \in P$ covers all points in the data space for which p is the nearest neighbors among the points from P.



Observations

- The Voronoi cells of two neighboring points $p_i, p_j \in P$ are separated by the perpendicular hyperplane („Mittelsenkrechte“) between p_i and p_j.
- As Voronoi cells are intersections of half spaces, they are convex regions.



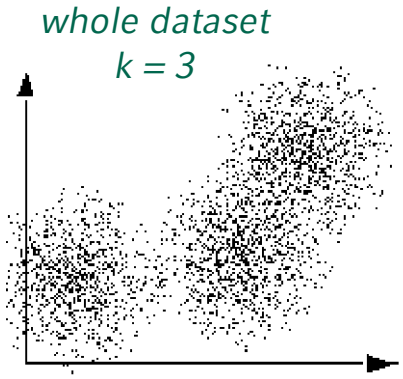
Contents

Initialization of Partitioning Clustering Methods

Just two examples:

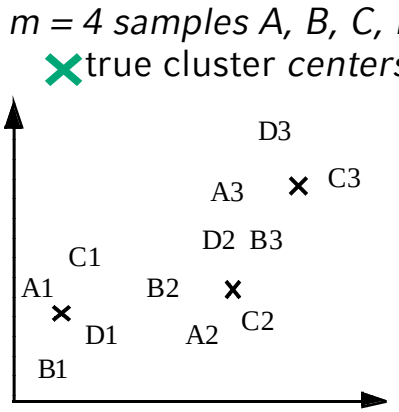
[naïve]

- Choose sample *A* of the dataset
- Cluster the sample and use centers as initialization



[Fayyad, Reina, and Bradley 1998]

- Choose *m* different (small) samples $A, \ldots, M$ of the dataset
- Cluster each sample to get *m* estimates for *k* representatives $A = (A_1, A_2, \ldots, A_k)$, $B = (B_1, \ldots, B_k)$, ..., $M = (M_1, \ldots, M_k)$
- Then, cluster the set $DS = A \cup B \cup \ldots \cup M$ *m* times. Each time use the centers of *A*, *B*, ..., *M* as respective initial partitioning
- Use the centers of the best clustering as initialization for the partitioning clustering of the whole dataset



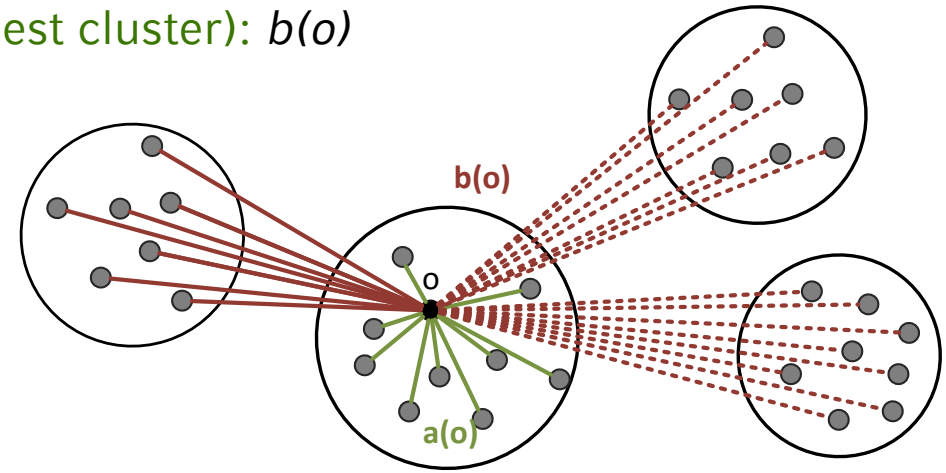
Fayyad U., Reina C., Bradley P. S., „Initialization of Iterative Refinement Clustering Algorithms“, *In KDD 1998)*, pp. 194—198.

Choice of the Parameter *k*

- Idea for a method:
  - Determine a clustering for each $k = 2, \ldots, n-1$
  - Choose the "best" clustering
- But how to measure the quality of a clustering?
  - A measure should not be monotonic over *k*.
  - The measures for the compactness of a clustering SSE and TD are monotonously decreasing with increasing value of *k*.
- Silhouette-Coefficient [Kaufman & Rousseeuw 1990]
  - Measure for the quality of a *k*-means or a *k*-medoid clustering that is not monotonic over *k*.

The Silhouette coefficient (1)

- Basic idea:
  - How good is the clustering = how appropriate is the mapping of objects to clusters
  - Elements in cluster should be „similar“ to their representative
    → measure the average distance of objects to their representative: *a(o)*
  - Elements in different clusters should be „dissimilar“
    → measure the average distance of objects to alternative clusters (i.e. second closest cluster): *b(o)*



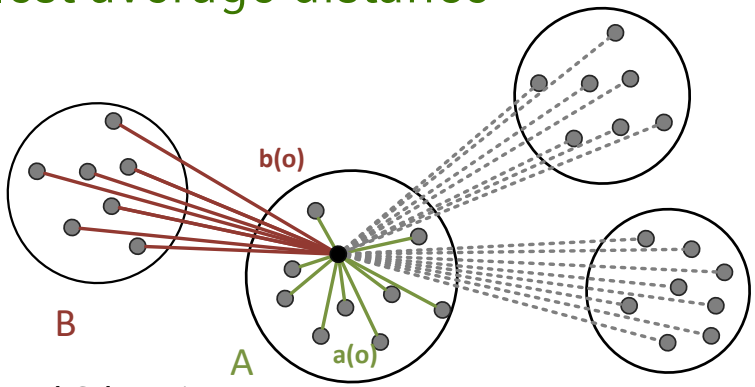
The Silhouette coefficient (2)

- $a(o)$: average distance between object o and the objects in its cluster A

$$a(o) = \frac{1}{|C(o)|} \sum_{p \in C(o)} dist(o,p)$$

- $b(o)$: for each other cluster C_i compute the average distance between o and the objects in C_i. Then take the smallest average distance

$$b(o) = \min_{C_i \neq C(o)} \left(\frac{1}{|C_i|} \sum_{p \in C_i} dist(o,p) \right)$$



- The silhouette of o is then defined as

$$s(o) = \begin{cases} 0 & if\ a(o) = 0, e.g. |C_i| = 1 \\ \dfrac{b(o) - a(o)}{\max\{a(o), b(o)\}} & else \end{cases}$$

- The values of the silhouette coefficient range from –1 to +1

The Silhouette coefficient (3)

- The silhouette of a cluster C_i is defined as:

$$silh(C_i) = \frac{1}{|C_i|} \sum_{o \in C_i} s(o)$$

- The silhouette of a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ is defined as:

$$silh(\mathcal{C}) = \frac{1}{|D|} \sum_{o \in D} s(o),$$

where D denotes the whole dataset.

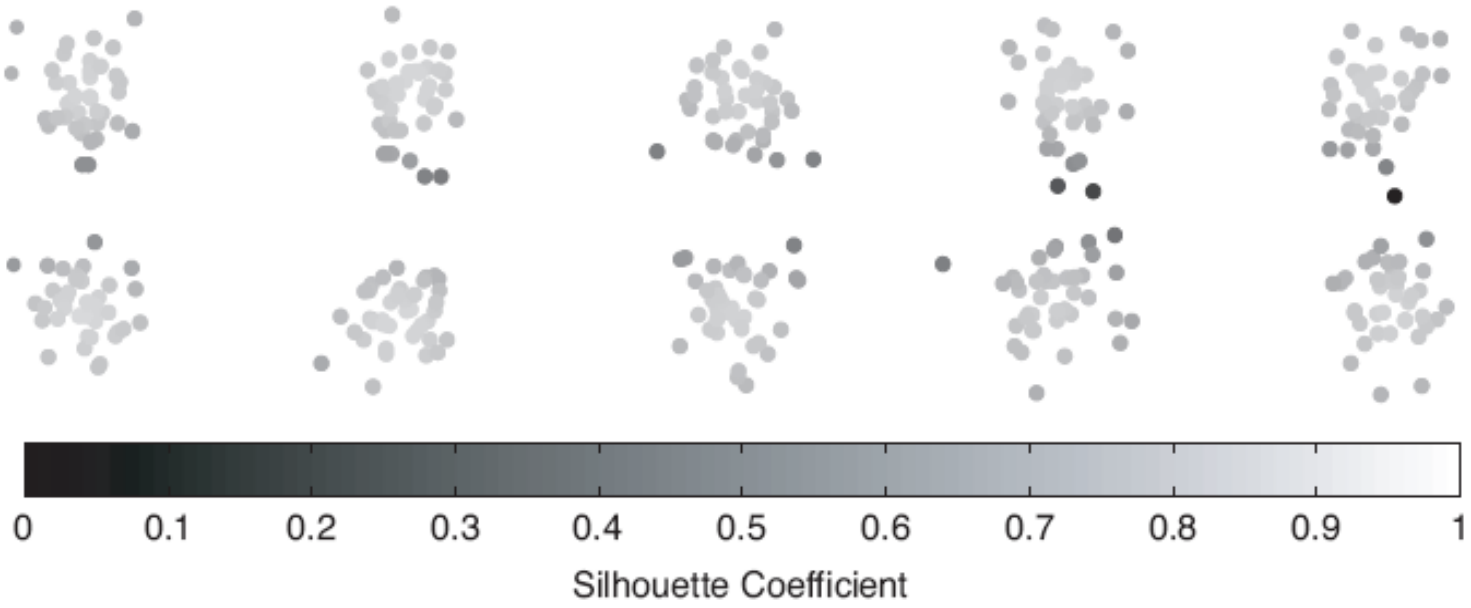
The Silhouette coefficient (4)

- „Reading“ the silhouette coefficient:
  Let $a(o) \neq 0$.
  - $b(o) \gg a(o) \Rightarrow s(o) \approx 1$: good assignment of o to its cluster A
  - $b(o) \approx a(o) \Rightarrow s(o) \approx 0$: o is in-between A and B
  - $b(o) \ll a(o) \Rightarrow s(o) \approx -1$: bad, on average o is closer to members of B

- Silhouette Coefficient s_C of a clustering: average silhouette of all objects
  - $0.7 < s_C \leq 1.0$ strong structure, $0.5 < s_C \leq 0.7$ medium structure
  - $0.25 < s_C \leq 0.5$ weak structure, $s_C \leq 0.25$ no structure

The Silhouette coefficient (5)

Silhouette Coefficient for points in ten clusters



in: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

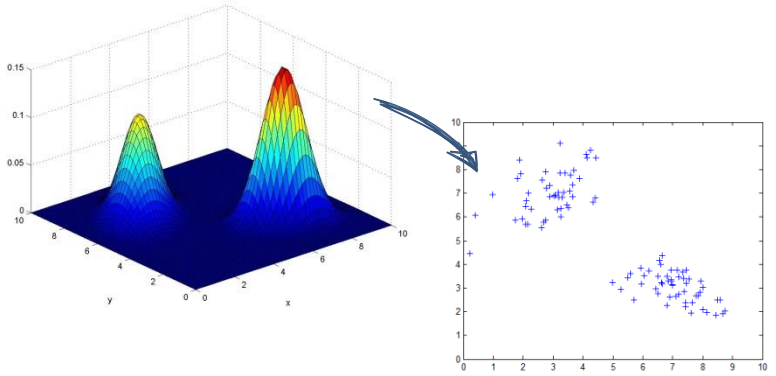
Contents

Expectation Maximization (EM)

Statistical approach for finding maximum likelihood estimates of parameters in probabilistic models

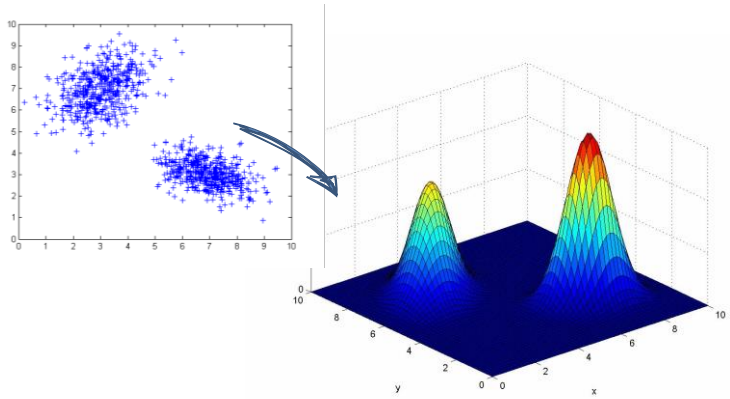
Here: using EM as clustering algorithm

Approach:
Observations are drawn from one of several components of a mixture distribution.



Main idea:

- Define clusters as probability distributions
  → each object has a certain probability of belonging to each cluster
- Iteratively improve the parameters of each distribution (e.g. center, "width" and "height" of a Gaussian distribution) until some quality threshold is reached



Additional Literature: C. M. Bishop „Pattern Recognition and Machine Learning", Springer, 2009

Excursus: Gaussian Mixture Distributions

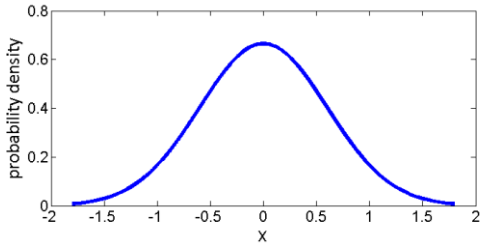
Note: EM is not restricted to Gaussian distributions, but they will serve as example in this lecture.

Gaussian distribution:

- Univariate: single variable x ∈ ℝ:

$$p(x|\mu,\sigma^2) = \mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{1}{2\sigma^2}\cdot(x-\mu)^2}$$

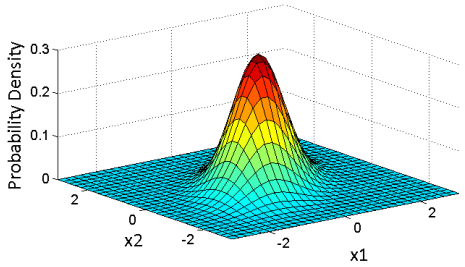
mean ∈ ℝ *variance* ∈ ℝ



- Multivariate: d-dimensional vector $\boldsymbol{x} \in \mathbb{R}^d$:

$$p(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} \cdot e^{-\frac{1}{2}\cdot(\boldsymbol{x}-\boldsymbol{\mu})^T\cdot(\boldsymbol{\Sigma})^{-1}\cdot(\boldsymbol{x}-\boldsymbol{\mu})}$$

mean vector ∈ $\mathbb{R}^d$ *covariance matrix* ∈ $\mathbb{R}^{d\times d}$

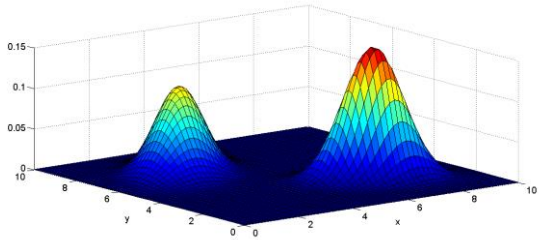


Gaussian mixture distribution with K components:

- d-dimensional vector $\boldsymbol{x} \in \mathbb{R}^d$:

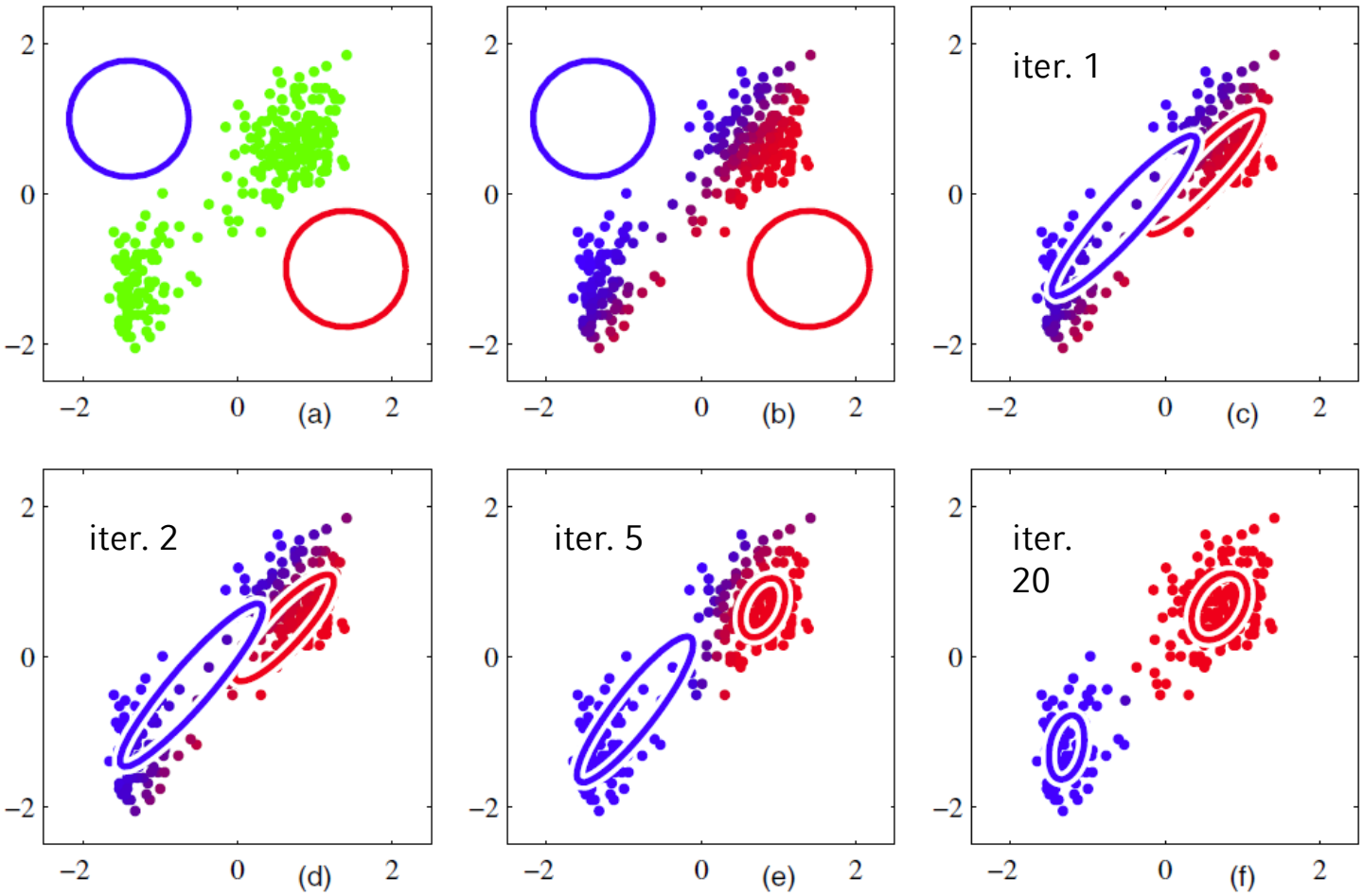
$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$$

mixing coefficients ∈ ℝ : $\sum_k \pi_k = 1$ *and* $0 \le \pi_k \le 1$



Expectation Maximization (EM): Exemplary Application

Example taken from: C. M. Bischop „Pattern Recognition and Machine Learning“, 2009



Expectation Maximization (EM)

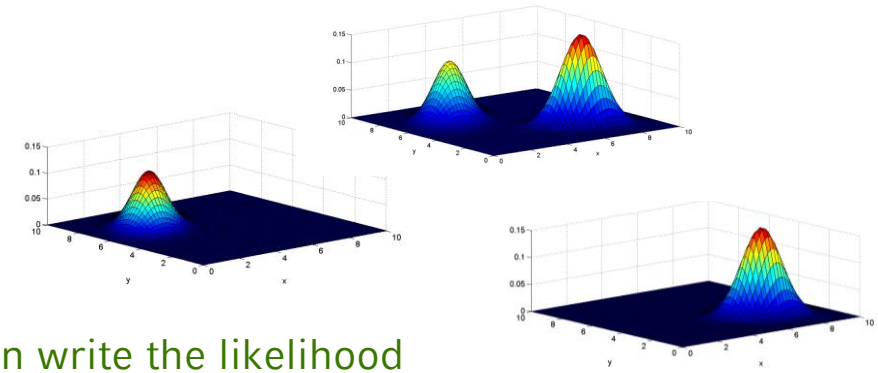
Note: EM is not restricted to Gaussian distributions, but they will serve as example in this lecture.

A *clustering* $\mathcal{M} = \{C_1, \dots, C_K\}$ is represented by a mixture distribution with parameters $\Theta = \{\pi_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \dots, \pi_K, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K\}$:

$$p(\boldsymbol{x}|\Theta) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

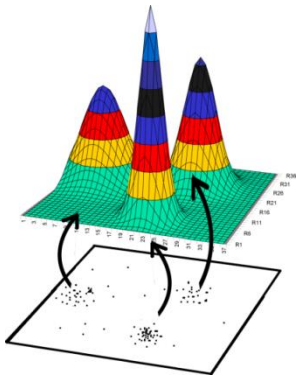
Each *cluster* is represented by one component of the mixture distribution:

$$p(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



Given a dataset $\mathbf{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subseteq \mathbb{R}^d$, we can write the likelihood that all data points $\mathbf{x}_n \in \mathbf{X}$ are generated (independently) by the mixture model with parameters Θ as:

$$\log p(\mathbf{X}|\Theta) = \log \prod_{n=1}^{N} p(x_n|\Theta)$$



Goal: Find the parameters Θ_{ML} with

maximal (log-)likelihood estimation (MLE)

$$\Theta_{ML} = \arg\max_{\Theta} \{\log p(\mathbf{X}|\Theta)\}$$

Expectation Maximization (EM)

- Goal: Find the parameters Θ_{ML} with the **maximal (log-)likelihood estimation**!

$$\Theta_{ML} = \arg\max_{\Theta}\{\log p(\mathbf{X}|\Theta)\}$$

$$\log p(\mathbf{X}|\Theta) = \log \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \cdot \mathrm{p}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \cdot \mathrm{p}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Maximization with respect to the means:

$$\frac{\partial}{\partial \boldsymbol{\mu}_j}\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_j)\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

$$\frac{\partial \log p(\mathbf{X}|\Theta)}{\partial \boldsymbol{\mu}_j} = \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n|\Theta)}{\partial \boldsymbol{\mu}_j} = \sum_{n=1}^{N} \frac{\frac{\partial p(\boldsymbol{x}_n|\Theta)}{\partial \boldsymbol{\mu}_j}}{p(\boldsymbol{x}_n|\Theta)} = \sum_{n=1}^{N} \frac{\frac{\partial\, \pi_j \cdot p(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\partial \boldsymbol{\mu}_j}}{\sum_{k=1}^{K} p(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} = \sum_{n=1}^{N} \frac{\pi_j \cdot \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_j)\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} p(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

$$\frac{\partial \log p(\mathbf{X}|\Theta)}{\partial \boldsymbol{\mu}_j} = \boldsymbol{\Sigma}_j^{-1} \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{\mu}_j) \frac{\pi_j \cdot \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \stackrel{\text{def}}{=} \mathbf{0}$$

- Define

$$\gamma_j(x_n) := \pi_j \cdot \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j).$$

$\gamma_j(x_n)$ is the probability that component j generated the object $\boldsymbol{x}_n$.

Expectation Maximization (EM)

Maximization w.r.t. the means yields:

$$\boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma_j(\boldsymbol{x}_n)\, \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_j(\boldsymbol{x}_n)} \qquad \text{(weighted mean)}$$

Maximization w.r.t. the covariance yields:

$$\boldsymbol{\Sigma}_j = \frac{\sum_{n=1}^{N} \gamma_j(\boldsymbol{x}_n)\left(\boldsymbol{x}_n - \boldsymbol{\mu}_j\right)\left(\boldsymbol{x}_n - \boldsymbol{\mu}_j\right)^T}{\sum_{n=1}^{N} \gamma_j(\boldsymbol{x}_n)}$$

Maximization w.r.t. the mixing coefficients yields:

$$\pi_j = \frac{\sum_{n=1}^{N} \gamma_j(\boldsymbol{x}_n)}{\sum_{k=1}^{K} \sum_{n=1}^{N} \gamma_k(\boldsymbol{x}_n)}$$

Expectation Maximization (EM)

Problem with finding the optimal parameters Θ_{ML}:

$$\boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma_j(\boldsymbol{x}_n)\, \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_j(\boldsymbol{x}_n)} \quad \text{and} \quad \gamma_j(\boldsymbol{x}_n) = \frac{\pi_j \cdot \mathcal{N}\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}$$

- Non-linear mutual dependencies.
- Optimizing the Gaussian of cluster j depends on all other Gaussians.

→ There is no closed-form solution!

→ Approximation through iterative optimization procedures

→ Break the mutual dependencies by optimizing $\boldsymbol{\mu}_j$ and $\gamma_j(\boldsymbol{x}_n)$ independently

Expectation Maximization (EM)

EM-approach: iterative optimization

1. Initialize means $\boldsymbol{\mu}_j$, covariances $\boldsymbol{\Sigma}_j$, and mixing coefficients π_j and evaluate the initial log likelihood.
2. **E step**: Evaluate the responsibilities using the current parameter values:

$$\gamma_j^{new}(\boldsymbol{x}_n) = \frac{\pi_j \cdot \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

3. **M step**: Re-estimate the parameters using the current responsibilities:

$$\boldsymbol{\mu}_j^{new} = \frac{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)\, \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)}$$

$$\boldsymbol{\Sigma}_j^{new} = \frac{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)\left(\boldsymbol{x}_n - \boldsymbol{\mu}_j^{new}\right)\left(\boldsymbol{x}_n - \boldsymbol{\mu}_j^{new}\right)^T}{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)}$$

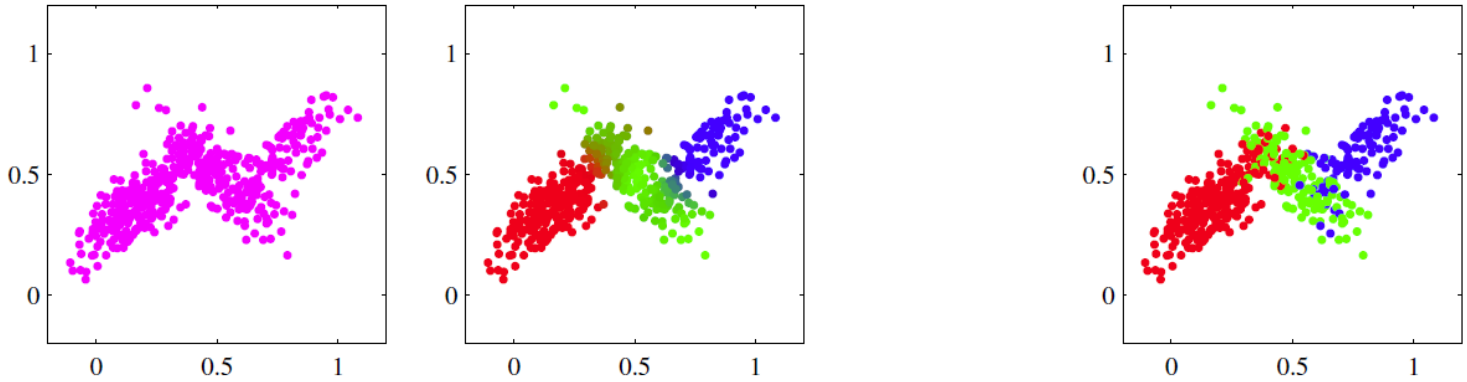
$$\pi_j^{new} = \frac{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)}{\sum_{k=1}^{K}\sum_{n=1}^{N} \gamma_k^{new}(\boldsymbol{x}_n)}$$

4. Evaluate the new log likelihood $\log p(\mathbf{X}|\Theta^{\text{new}})$ and check for convergence of parameters or log likelihood ($|\log p(\mathbf{X}|\Theta^{\text{new}}) - \log p(\mathbf{X}|\Theta)| \leq \epsilon$).
   If the convergence criterion is not satisfied, set $\Theta = \Theta^{\text{new}}$ and go to step 2.

EM: Turning the Soft Clustering into a Partitioning

EM obtains a *soft* clustering (each object belongs to each cluster with a certain probability) reflecting the uncertainty of the most appropriate assignment.

Example taken from: C. M. Bishop „Pattern Recognition and Machine Learning“, 2009



a) input for EM *b) soft clustering result of EM* *c) original data*

Modification to obtain a *partitioning* variant

- Assign each object to the cluster to which it belongs with the highest probability

$$\text{Cluster}(\text{object}_n) = argmax_{k \in \{1,\ldots,K\}}\{\gamma(z_{nk})\}$$

Discussion

Superior to k-Means for clusters of varying size
or clusters having differing variances
→ more accurate data representation

Convergence to (possibly local) maximum

Computational effort for N objects, K derived clusters, and t iterations:

- $O(t \cdot N \cdot K)$
- #iterations is quite high in many cases

Both - result and runtime - strongly depend on

- the initial assignment

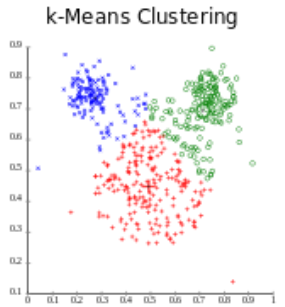
  → do multiple random starts and choose the final estimate with highest likelihood

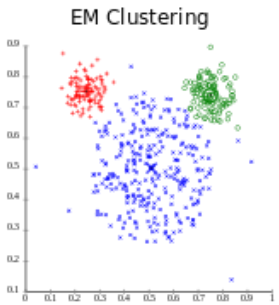
  → Initialize with clustering algorithms (e.g., K-Means usually converges much faster)

  → Local maxima and initialization issues have been addressed in various extensions of EM

- a proper choice of parameter K (= desired number of clusters)

  → Apply principals of model selection (see next slide)





EM: Model Selection for Determining Parameter K

Classical trade-off problem for selecting the proper number of components K

- If K is too high, the mixture may overfit the data
- If K is too low, the mixture may not be flexible enough to approximate the data

Idea: determine candidate models Θ_K for a range of values of K (from K_{min} to K_{max}) and select the model $\Theta_{K^*} = \max\{\text{qual}(\Theta_K) | K \in \{K_{min}, \ldots, K_{max}\}\}$

- Silhouette Coefficient (as for k-Means) only works for partitioning approaches.
- The MLE (Maximum Likelihood Estimation) criterion is nondecreasing in K

Solution: deterministic or stochastic *model selection* methods[MP'00] which try to balance the goodness of fit with simplicity.

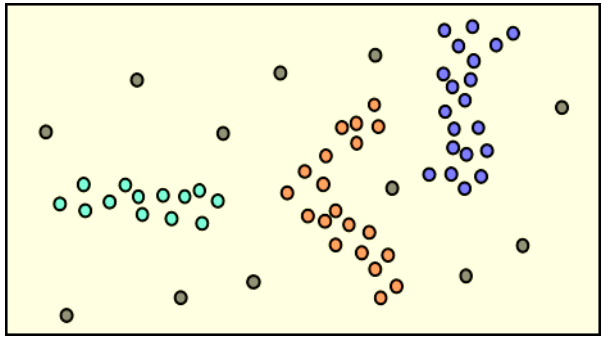
- Deterministic: $qual(\Theta_K) = \log p(\mathbf{X}|\Theta_K) + \mathcal{P}(K)$
  where $\mathcal{P}(K)$ is an increasing function penalizing higher values of K
- Stochastic: based on Markov Chain Monte Carlo (MCMC)

[MP'00] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, New York, 2000.

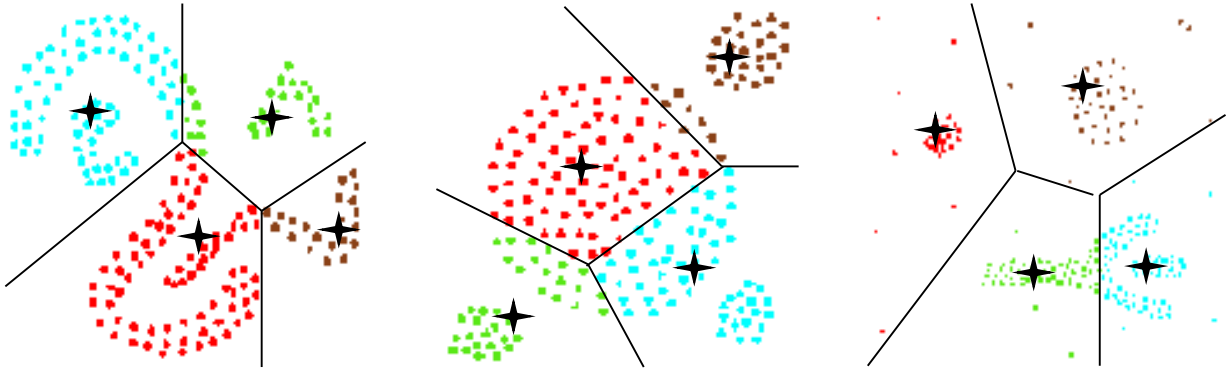
Contents

Density-Based Clustering

- Basic Idea:
  - Clusters are dense regions in the data space, separated by regions of lower object density



- Why Density-Based Clustering?



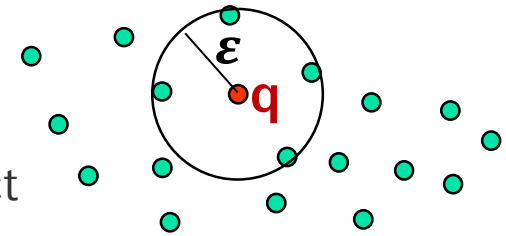
Results of a *k*-medoid algorithm for *k*=4

- Different density-based approaches exist (see Textbook & Papers)
  Here we discuss the ideas underlying the DBSCAN algorithm

Density-Based Clustering: Basic Concept

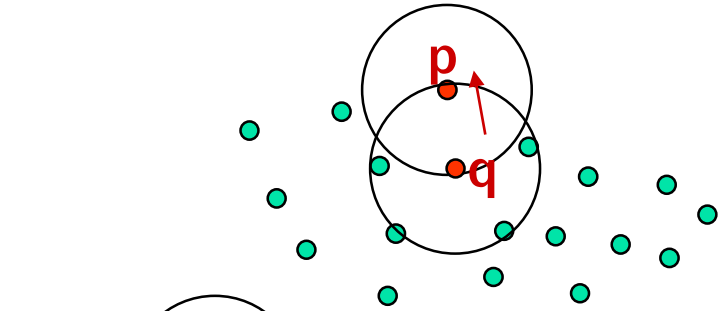
- Intuition for the formalization of the basic idea
  - For any point in a cluster, the local point density around that point has to exceed some threshold
  - The set of points from one cluster is spatially connected
- Local point density at a point q defined by two parameters
  - ε–radius for the neighborhood of point q:
    $N_\varepsilon(q) := \{p \in D | dist(p,q) \leq \varepsilon\}$ *! contains q itself !*
  - ***MinPts*** – minimum number of points in the given neighbourhood $N_\varepsilon(q)$
- q is called a ***core object*** (or core point) w.r.t. ε, *MinPts* if $| N_\varepsilon(q) | \geq MinPts$

MinPts = 5 → **q** is a core object

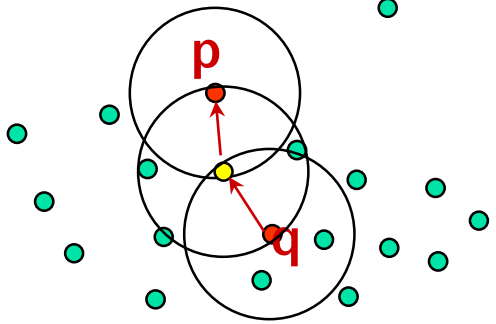


Density-Based Clustering: Basic Definitions

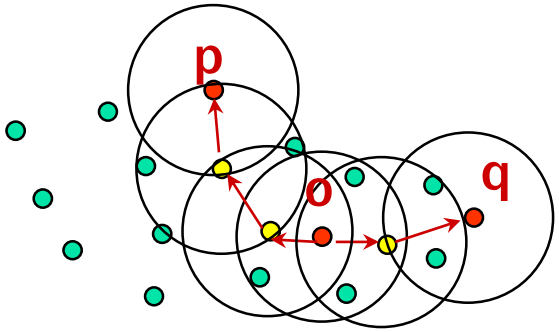
- *p* ***directly density-reachable*** from *q* w.r.t. ε, *MinPts* if
  1) $p \in N_\varepsilon(q)$ and
  2) *q* is a core object w.r.t. ε, *MinPts*



- ***density-reachable***: transitive closure of *directly* density-reachable



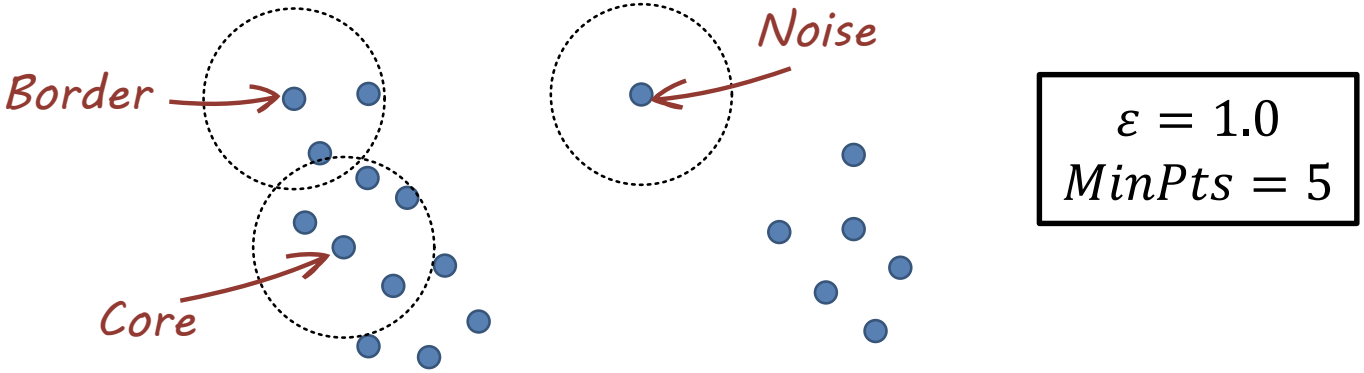
- *p* is ***density-connected*** to a point *q* w.r.t. ε, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. ε, *MinPts*.



Density-Based Clustering: Basic Definitions

- ***Density-Based Cluster:*** non-empty subset S of database D satisfying:
  1) *Maximality*: if p is in S and q is density-reachable from p then q is in S
  2) *Connectivity*: each object in S is density-connected to all other objects in S

- ***Density-Based Clustering*** of a database D : $\{S_1, ..., S_n; N\}$ where
  - $S_1, ..., S_n$: all density-based clusters in the database D
  - $N = D \setminus \{S_1 \cup ... \cup S_n\}$ is called the ***noise*** (objects not in any cluster)



$\varepsilon = 1.0$
$MinPts = 5$

Density-Based Clustering: DBSCAN Algorithm

- Density Based Spatial Clustering of Applications with Noise
- Basic Theorem:
  - Each object in a density-based cluster C is density-reachable from any of its core-objects
  - Nothing else is density-reachable from core objects.

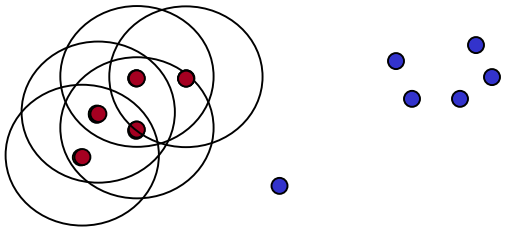
```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

  - density-reachable objects are collected by performing successive ε-neighborhood queries.

Ester M., Kriegel H.-P., Sander J., Xu X.: „A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise“, *In KDD 1996*, pp. 226—231.

DBSCAN Algorithm: Example

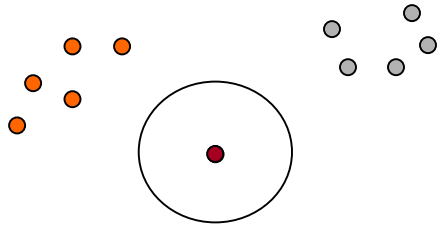
- Parameter
  - $\varepsilon = 2.0$
  - *MinPts* = 3



for each $o \in D$ **do**
  if o is not yet classified **then**
    if o is a core-object **then**
      collect all objects density-reachable from o and assign them to a new cluster.
    else
      assign o to NOISE

DBSCAN Algorithm: Example

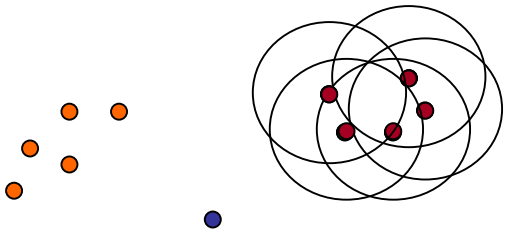
- Parameter
  - $\varepsilon = 2.0$
  - $MinPts = 3$



for each $o \in D$ **do**
  if o is not yet classified **then**
    if o is a core-object **then**
      collect all objects density-reachable from o
      and assign them to a new cluster.
    else
      assign o to NOISE

DBSCAN Algorithm: Example

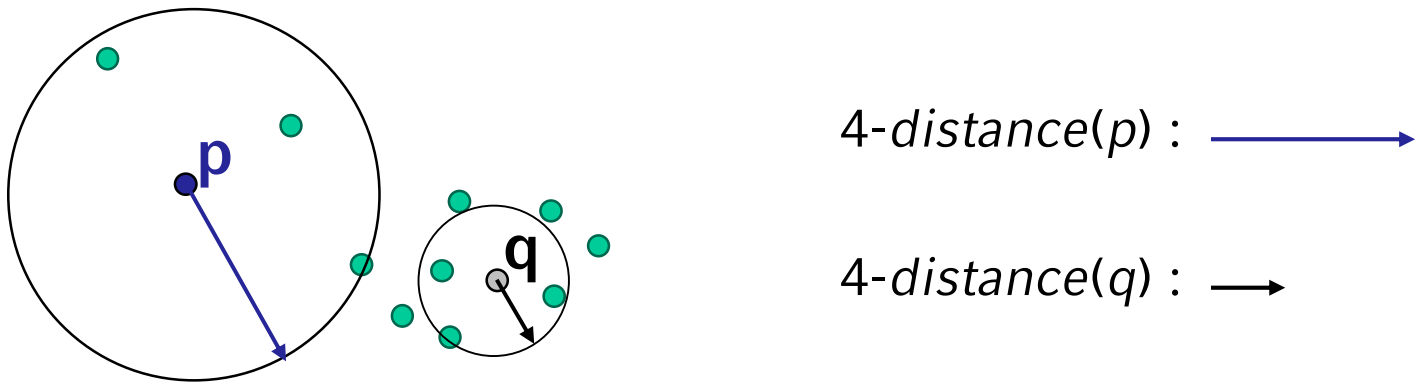
- Parameter
  - $\varepsilon = 2.0$
  - *MinPts* = 3



for each $o \in D$ **do**
  if o is not yet classified **then**
    if o is a core-object **then**
      collect all objects density-reachable from o and assign them to a new cluster.
    else
      assign o to NOISE

Determining the Parameters ε and *MinPts*

- Cluster: Point density higher than specified by ε and *MinPts*
- Idea: use the point density of the least dense cluster in the data set as parameters – but how to determine this?
- Heuristic: look at the distances to the *k*-nearest neighbors

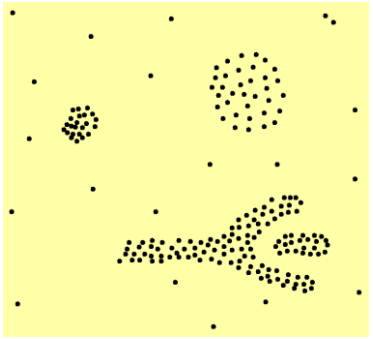


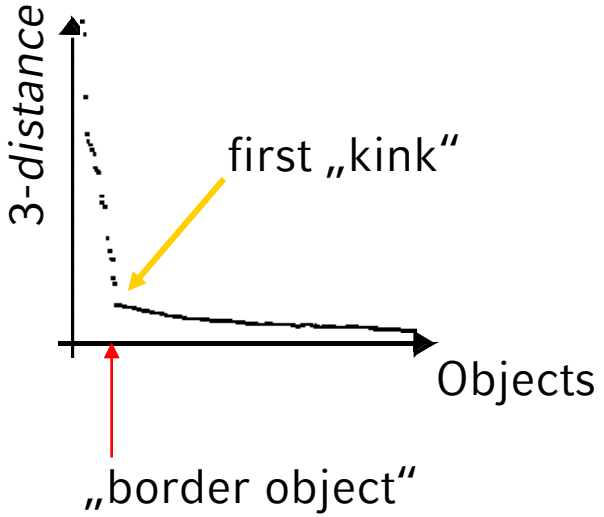
- Function *k-distance*(*p*): distance from *p* to the its *k*-nearest neighbor
- *k-distance plot*: *k*-distances of all objects, sorted in decreasing order

Determining the Parameters ε and *MinPts*

- Heuristic method:
  - Fix a value for *MinPts*
  - (default: $2 \times d - 1$, d = dimension of data space)
  - User selects "border object" *o* from the *MinPts-distance* plot; ε is set to *MinPts-distance*(o)

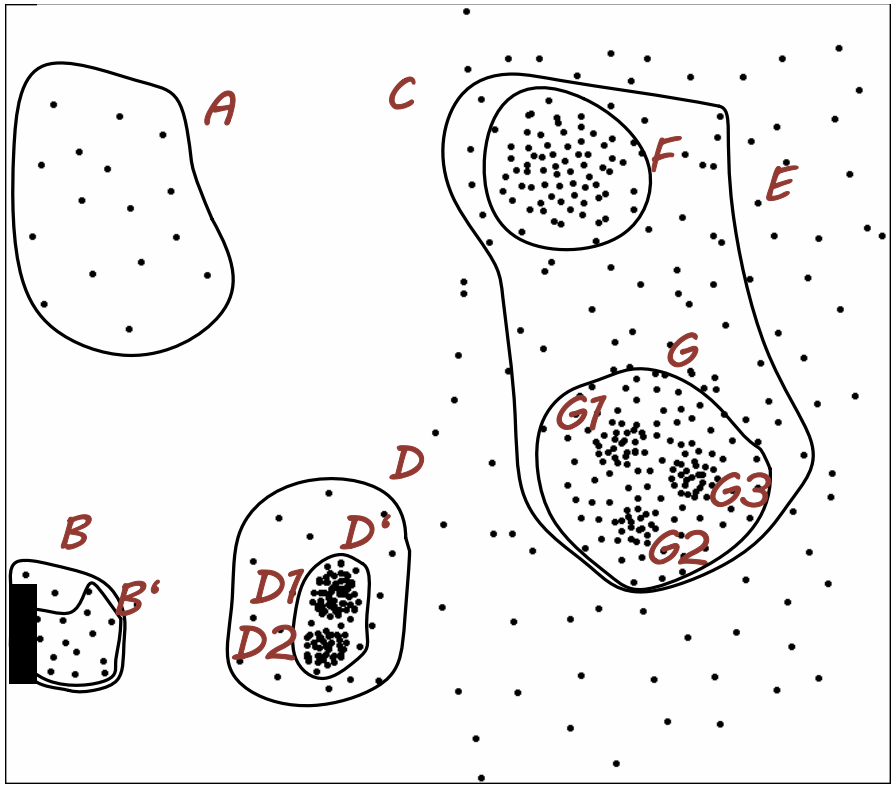
- Example *k*-distance plot
  1. dim = 2 → MinPts = 3
  2. Identify border object (kink)
  3. Set ε

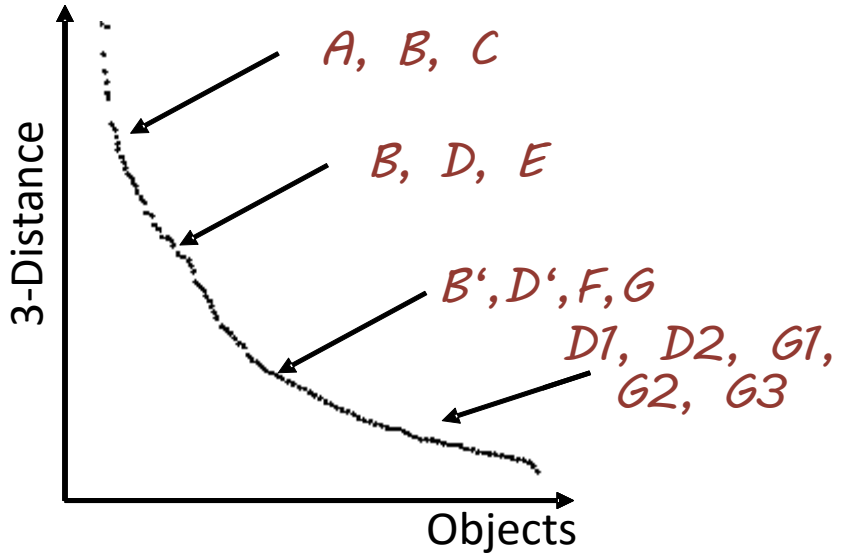




Determining the Parameters ε and *MinPts*

Problematic example





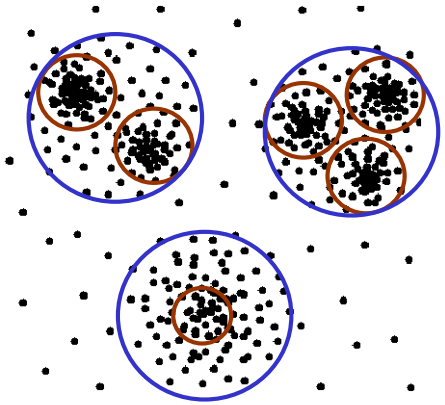
Density-Based Clustering: Discussion

- Advantages
  - Clusters can have arbitrary shape and size, i.e. clusters are not restricted to have convex shapes
  - Number of clusters is determined automatically
  - Can separate clusters from surrounding noise
  - Can be supported by spatial index structures
  - Complexity: N_ε-query: O(n) DBSCAN: O(n^2)
- Disadvantages
  - Input parameters may be difficult to determine
  - In some situations very sensitive to input parameter setting

Contents

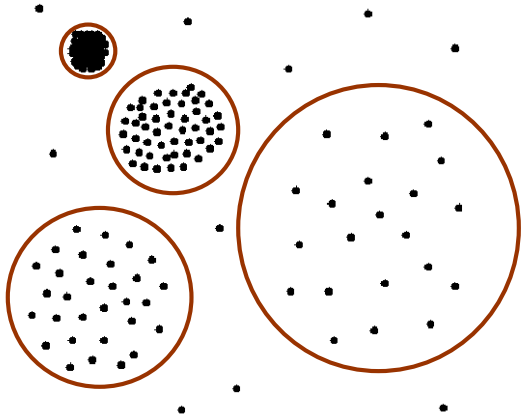
From Partitioning to Hierarchical Clustering

- Global parameters to separate all clusters with a partitioning clustering method may not exist



hierarchical cluster structure

and/or

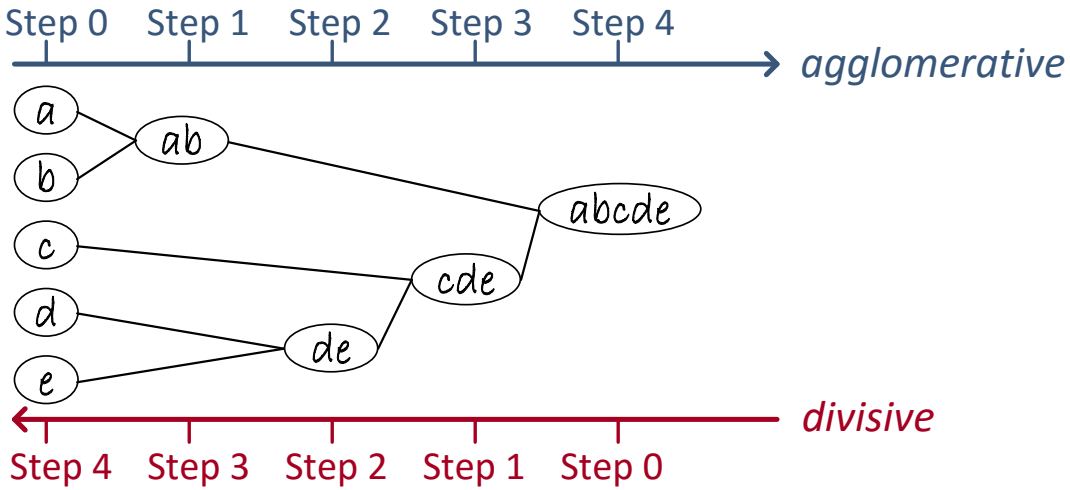


largely differing densities and sizes

- Need a hierarchical clustering algorithm in these situations

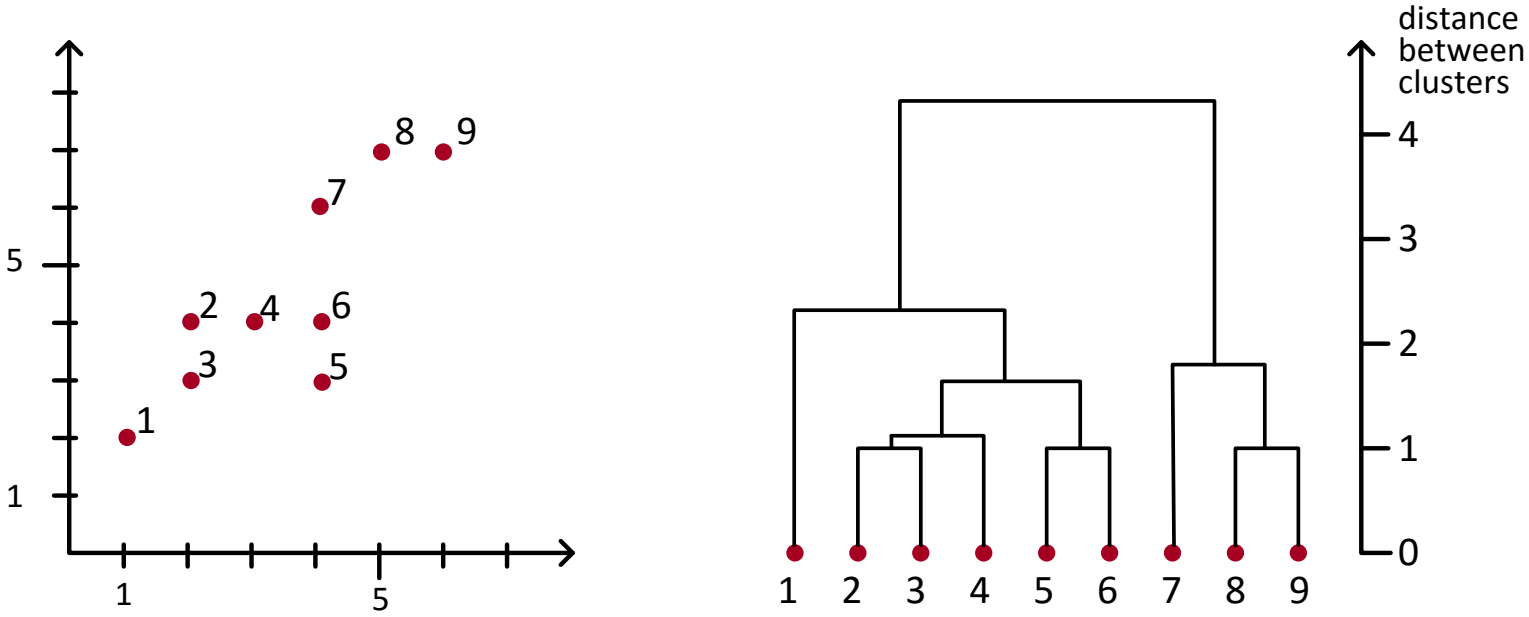
Hierarchical Clustering: Basic Notions

- Hierarchical decomposition of the data set (with respect to a given similarity measure) into a set of nested clusters
- Result represented by a so called *dendrogram* (greek δενδρο = tree)
  - Nodes in the dendrogram represent possible clusters
  - can be constructed bottom-up (agglomerative approach) or top down (divisive approach)



Hierarchical Clustering: Example

- Interpretation of the dendrogram
  - The root represents the whole data set
  - A leaf represents a single object in the data set
  - An internal node represents the union of all objects in its sub-tree
  - The height of an internal node represents the distance between its two child nodes



Contents

Agglomerative Hierarchical Clustering

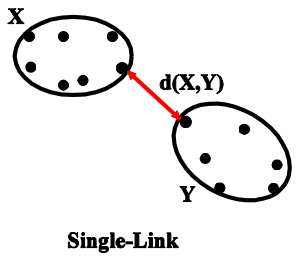
1. Initially, each object forms its own cluster
2. Consider all pairwise distances between the initial clusters (objects)
3. Merge the closest pair (A, B) in the set of the current clusters into a new cluster $C = A \cup B$
4. Remove A and B from the set of current clusters; insert C into the set of current clusters
5. If the set of current clusters contains only C (i.e., if C represents all objects from the database): STOP
6. Else: determine the distance between the new cluster C and all other clusters in the set of current clusters; go to step 3.

- Requires a distance function for clusters (sets of objects)

Single Link Method and Variants

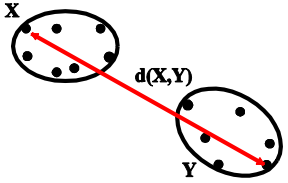
- Given: a distance function *dist*(*p*, *q*) for database objects
- The following distance functions for clusters (i.e., sets of objects) *X* and *Y* are commonly used for hierarchical clustering:

Single-Link: $$dist\_sl(X,Y) = \min_{x \in X, y \in Y} dist(x,y)$$



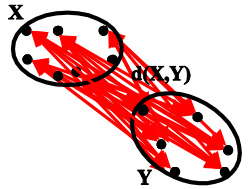
Single-Link

Complete-Link: $$dist\_cl(X,Y) = \max_{x \in X, y \in Y} dist(x,y)$$



Complete-Link

Average-Link: $$dist\_al(X,Y) = \frac{1}{|X| \cdot |Y|} \cdot \sum_{x \in X, y \in Y} dist(x,y)$$



Average-Link

Divisive Hierarchical Clustering

General approach: Top Down

- Initially, all objects form one cluster
- Repeat until all objects are singletons
  - Choose a cluster to split ⇐ **how ?**
  - Replace the chosen cluster with the sub-clusters ⇐ **how to split ?** e.g., 'reversing' agglomerative approach and split into two

Example solution: DIANA

- Select the cluster C with largest diameter for splitting
- Search the most disparate observation o in C (highest average dissimilarity)
  - $SplinterGroup := \{o\}$
  - Iteratively assign the $o' \in C \backslash SplinterGroup$ with the highest $D(o') > 0$ to the splinter group until for all $o' \in C \backslash SplinterGroup$: $D(o') \leq 0$

$$D(o') = \sum_{o_j \in C \backslash SplinterGroup} \frac{d(o', o_j)}{|C \backslash SplinterGroup|} - \sum_{o_i \in SplinterGroup} \frac{d(o', o_i)}{|SplinterGroup|}$$

Discussion Agglomerative vs. Divisive HC

Divisive HC and Agglomerative HC need n-1 steps

- Agglomerative HC has to consider $\frac{n \cdot (n-1)}{2} = \binom{n}{2}$ combinations in the first step
- Divisive HC has $2^{n-1} - 1$ many possibilities to split the data in its first step

  →Not every possibility has to be considered (DIANA)

Divisive HC is conceptually more complex since it needs a second 'flat' clustering algorithm (splitting procedure)

Agglomerative HC decides based on local patterns

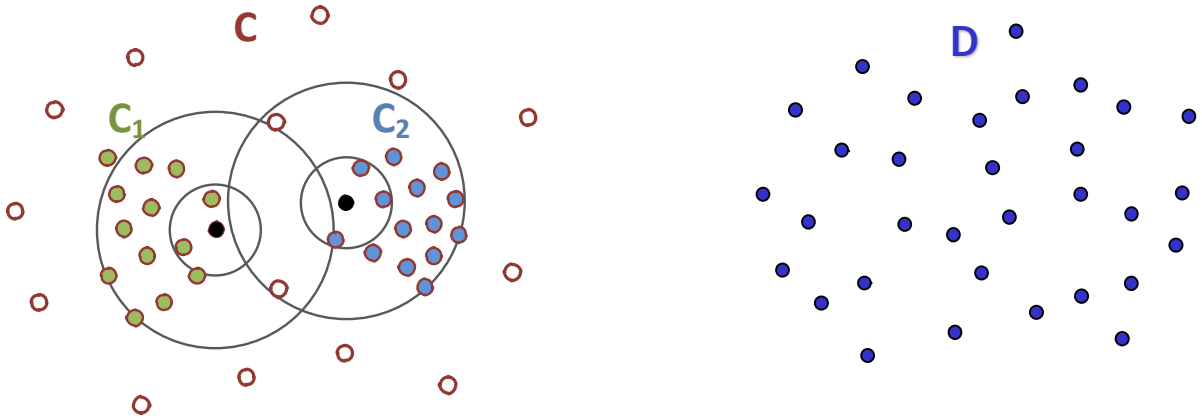
Divisive HC uses complete information about the global data distribution

→more accurate than Agglomerative HC?!

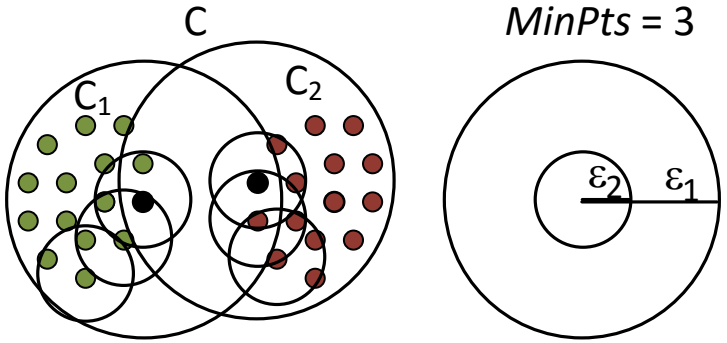
Contents

Density-Based Hierarchical Clustering

- *Observation*: Dense clusters are completely contained by less dense clusters



- *Idea*: Process objects in the "right" order and keep track of point density in their neighborhood

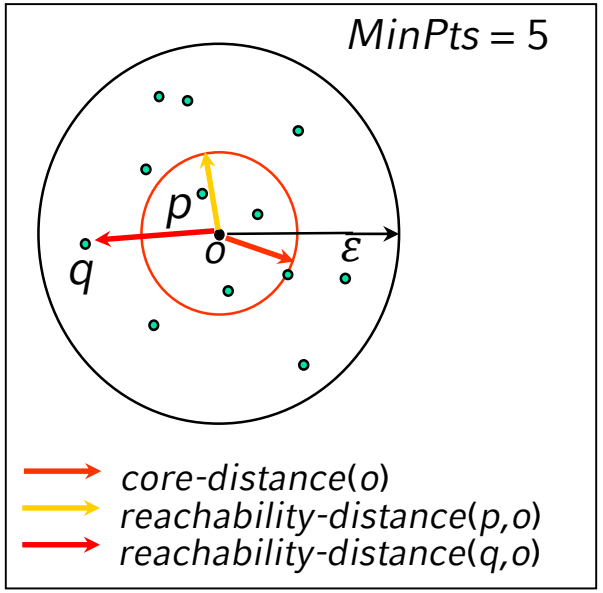


Core Distance and Reachability Distance

- Parameters: "generating" distance ε, fixed value *MinPts*
- ***core-distance***$_{\varepsilon,MinPts}$***(o)***

  "smallest distance such that *o* is a core object"
  (if core-distance > ε: "?")
- ***reachability-distance***$_{\varepsilon,MinPts}$***(p, o)***

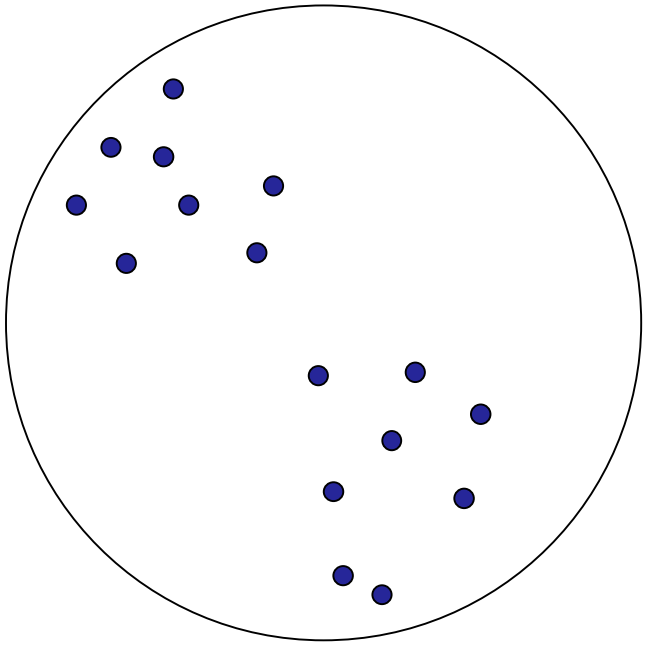
  "smallest distance such that *p* is *directly* density-reachable from *o*"
  (if reachability-distance > ε: ∞)



$$reach\_dist(p,o) = \begin{cases} dist(p,o) & ,dist(p,o) \geq core\_dist(o) \\ core\_dist(o) & ,dist(p,o) < core\_dist(o) \\ \infty & ,dist(p,o) > \epsilon \end{cases}$$

The Algorithm OPTICS

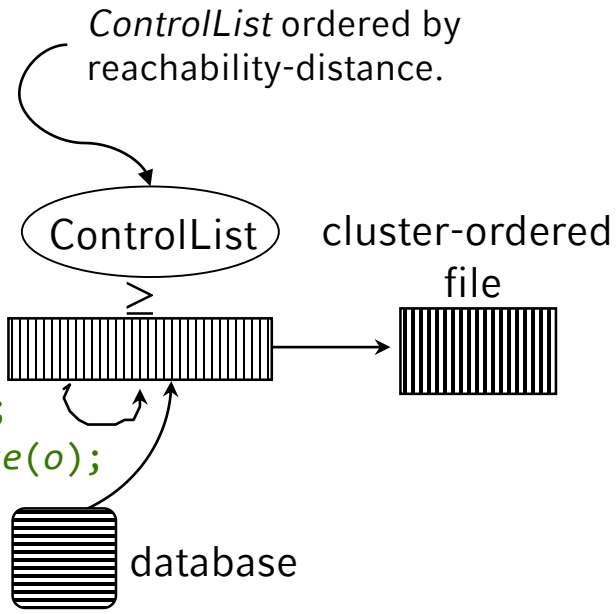
- OPTICS: **O**rdering **P**oints **T**o **I**dentify the **C**lustering **S**tructure
- Basic data structure: controlList
  - Memorize shortest reachability distances seen so far ("distance of a jump to that point")
- Visit each point
  - Make always a shortest jump
- Output:
  - order of points
  - core-distance of points
  - reachability-distance of points



Ankerst M., Breunig M., Kriegel H.-P., Sander J.: „OPTICS: Ordering Points To Identify the Clustering Structure", *In SIGMOD 1999,* pp. 49-60.

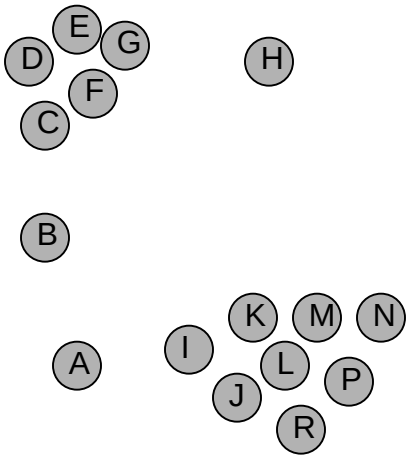
The Algorithm OPTICS

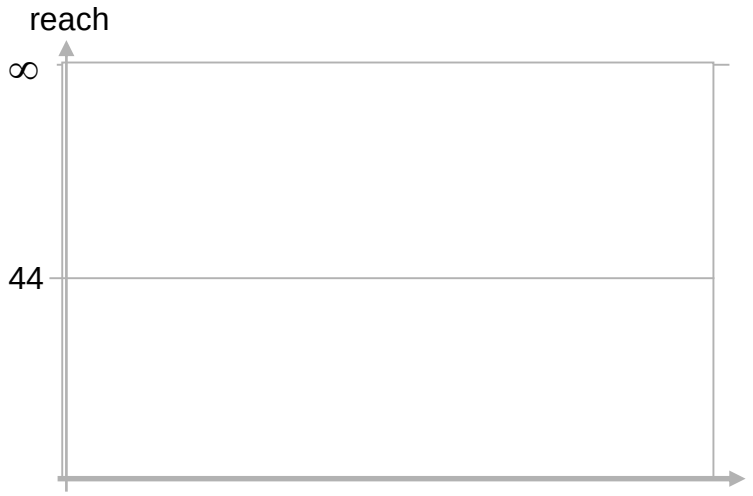

```
foreach o ∈ Database
  // initially, o.processed = false for all objects o

  if o.processed = false;
     insert (o, ∞) into ControlList;

  while ControlList contains objects not yet processed
        select first element (o, r-dist) from ControlList;
        retrieve N_e(o) and determine c_dist = core-distance(o);
        set o.processed = true;
        write (o, r_dist, c_dist) to file;
        if o is a core object at any distance £ e
          foreach p Î N_e(o) not yet processed;
                determine r_dist_p = reachability-distance(p, o);
                if (p, _) Ï ControlList
                    insert (p, r_dist_p) in ControlList;
                else if (p, old_r_dist) Î ControlList and r_dist_p < old_r_dist
                    update (p, r_dist_p) in ControlList;
```



The Algorithm OPTICS - Example

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

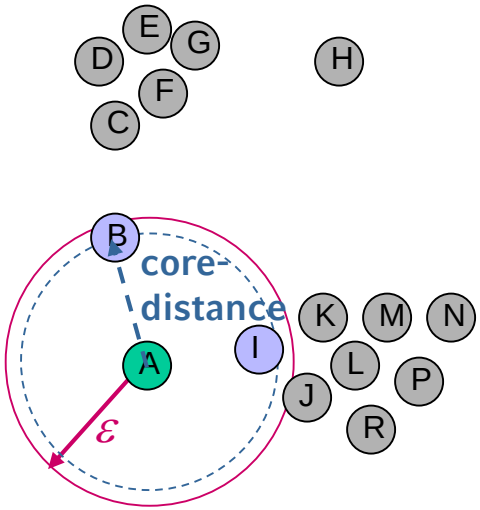


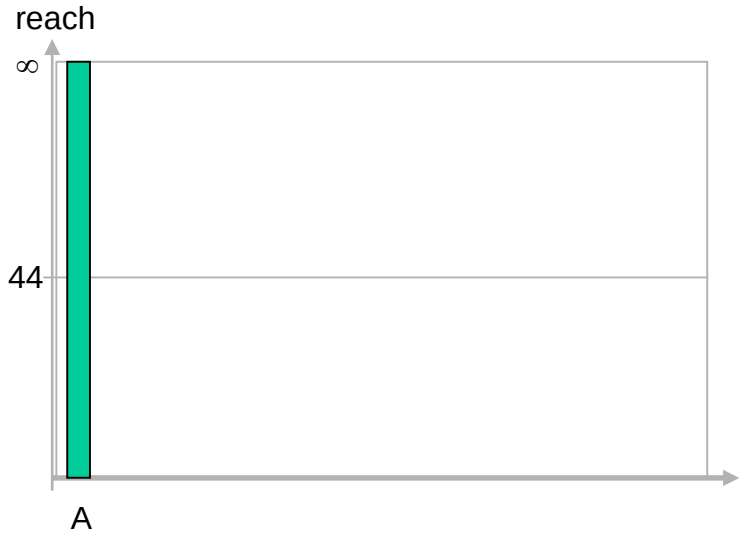


seed list:

The Algorithm OPTICS – Example (2)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

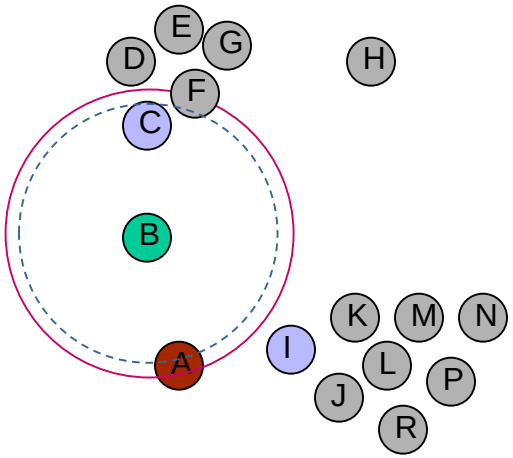


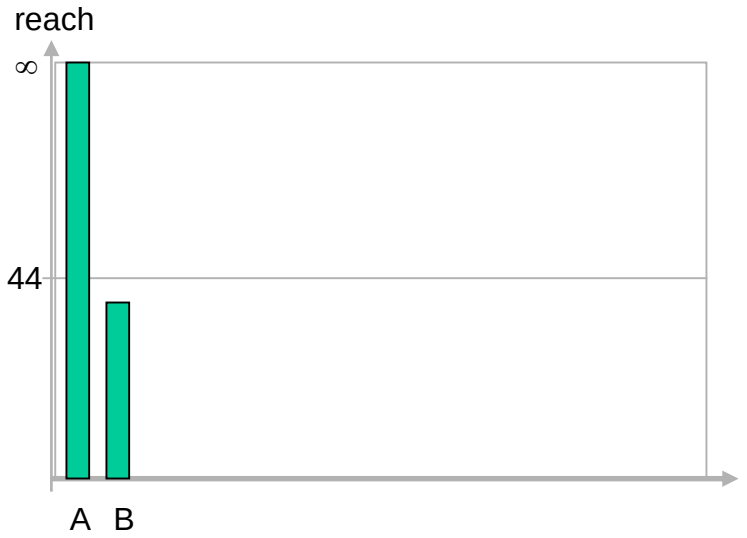


seed list: (B,40) (I, 40)

The Algorithm OPTICS – Example (3)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

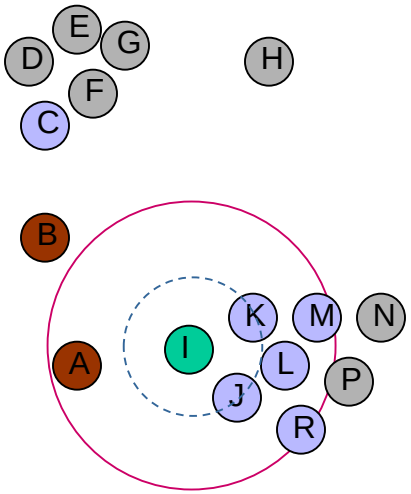




seed list: (I, 40) (C, 40)

The Algorithm OPTICS – Example (4)

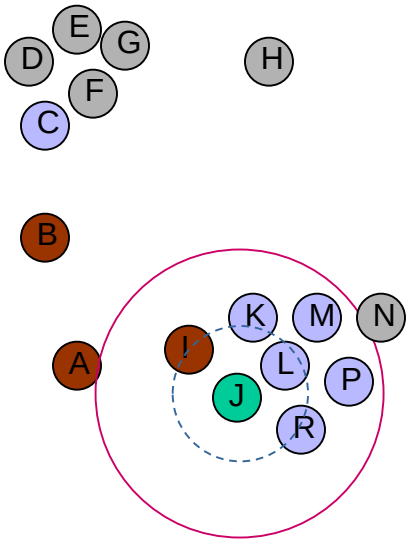
- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$



seed list: (J, 20) (K, 20) (L, 31) (C, 40) (M, 40) (R, 43)

The Algorithm OPTICS – Example (5)

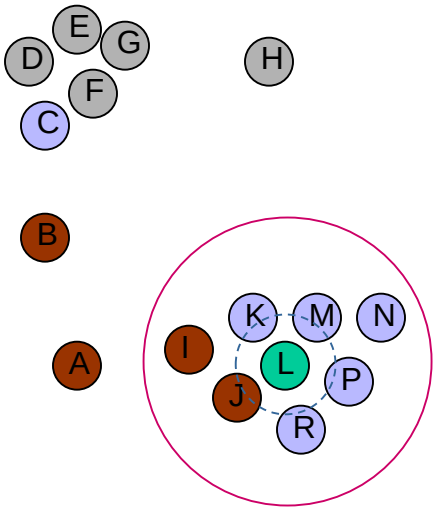
- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

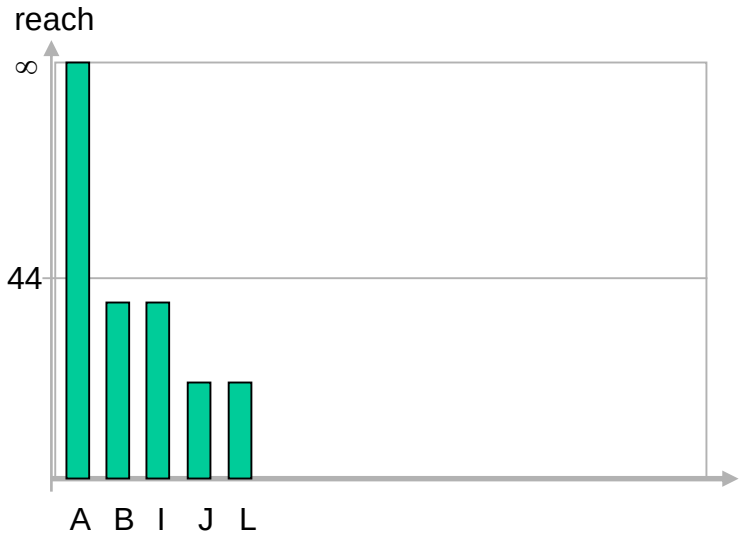


seed list: (L, 19) (K, 20) (R, 21) (M, 30) (P, 31) (C, 40)

The Algorithm OPTICS – Example (6)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

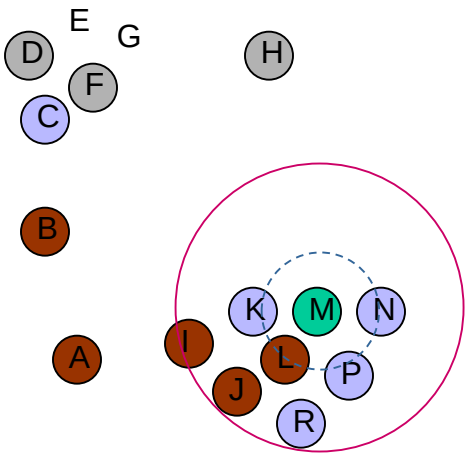


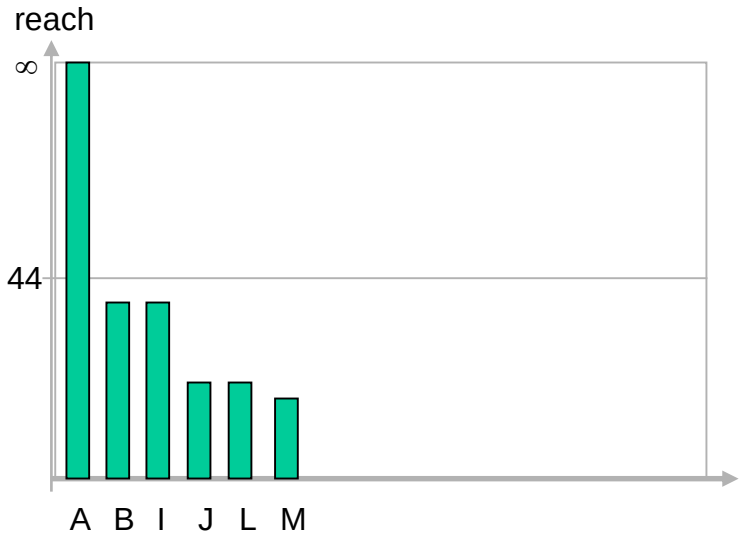


seed list: (M, 18) (K, 18) (R, 20) (P, 21) (N, 35) (C, 40)

The Algorithm OPTICS – Example (7)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

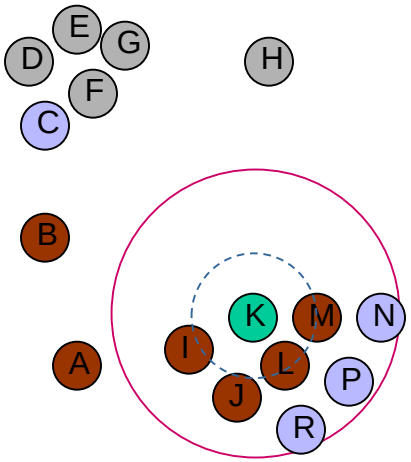


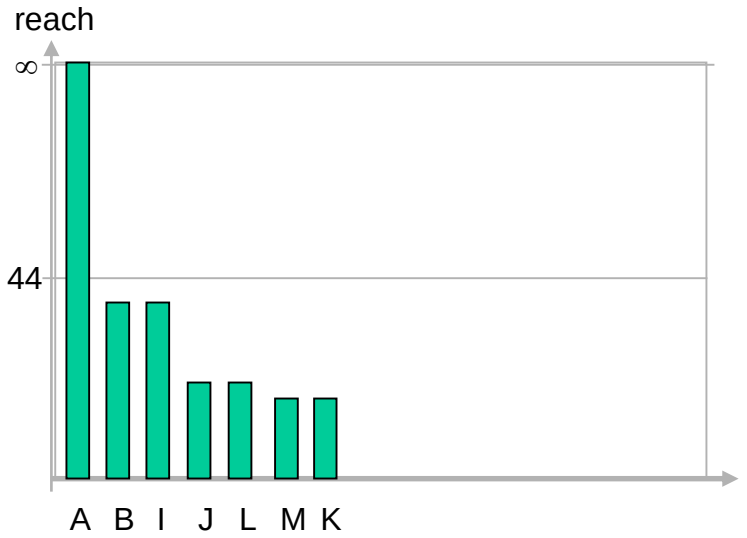


seed list: (K, 18) (N, 19) (R, 20) (P, 21) (C, 40)

The Algorithm OPTICS – Example (8)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

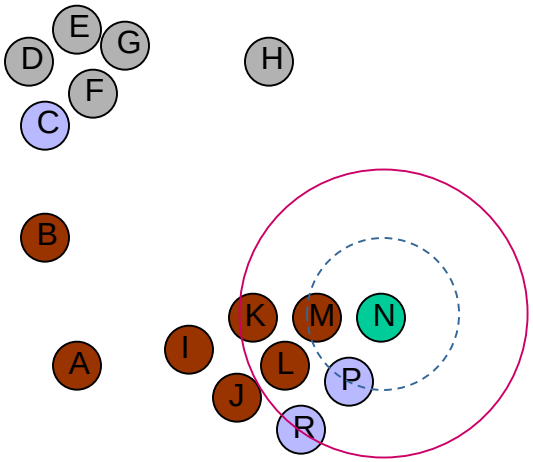


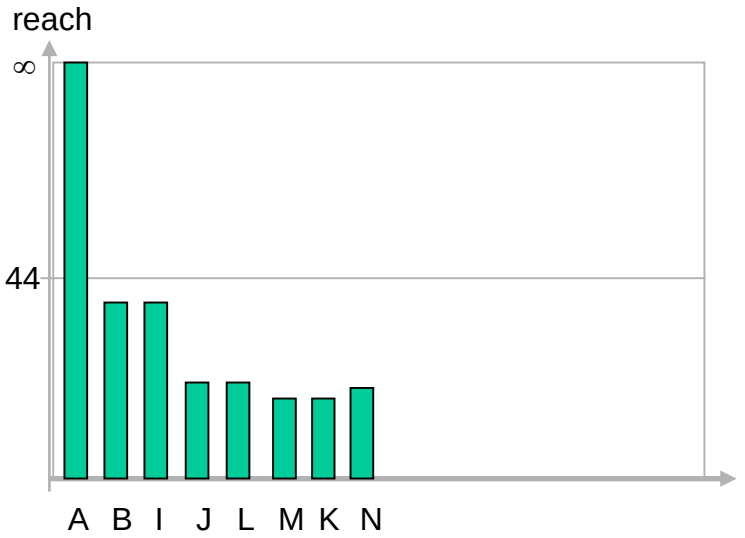


seed list: (N, 19) (R, 20) (P, 21) (C, 40)

The Algorithm OPTICS – Example (9)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

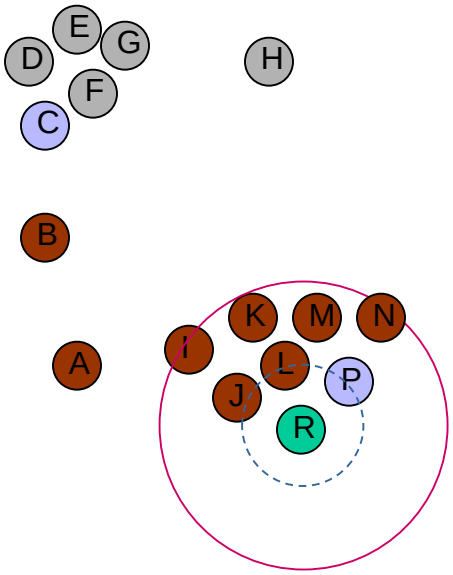


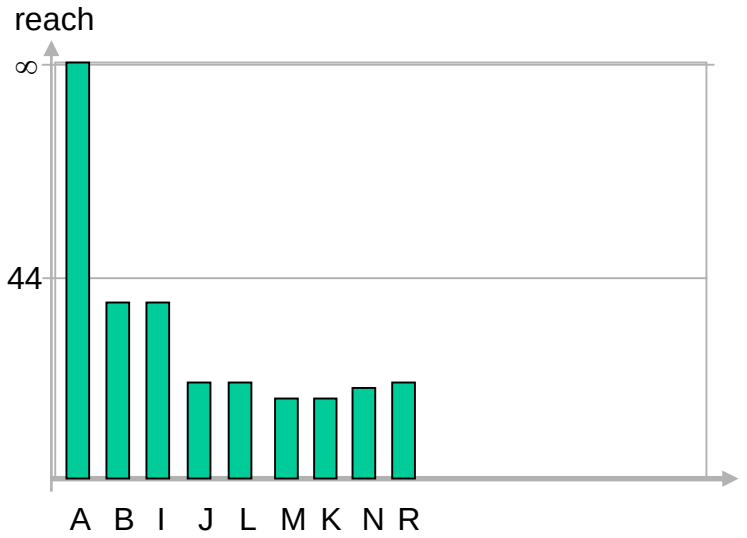


seed list: (R, 20) (P, 21) (C, 40)

The Algorithm OPTICS – Example (10)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

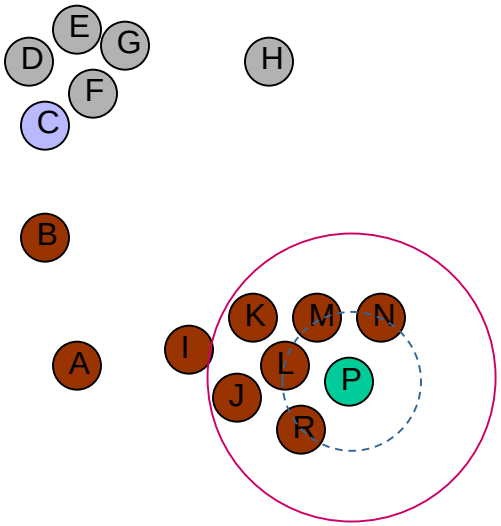




seed list: (P, 21) (C, 40)

The Algorithm OPTICS – Example (11)

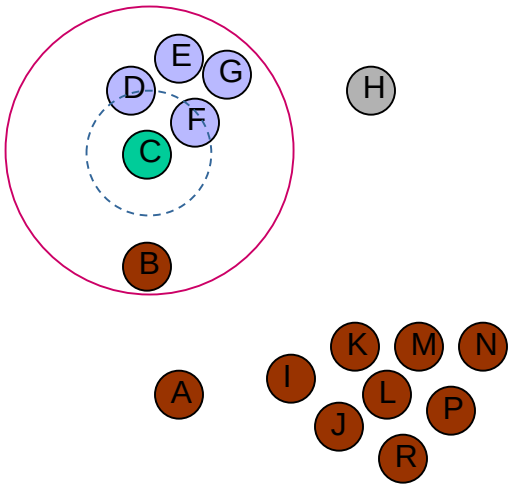
- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

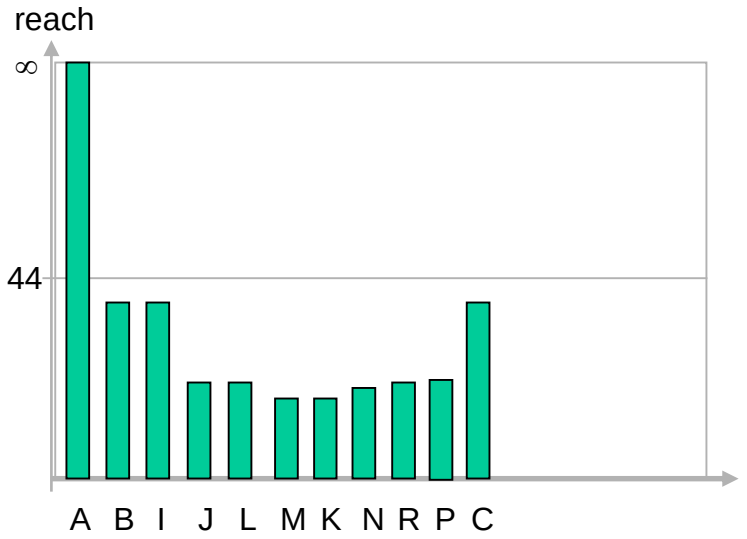


seed list: (C, 40)

The Algorithm OPTICS – Example (12)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

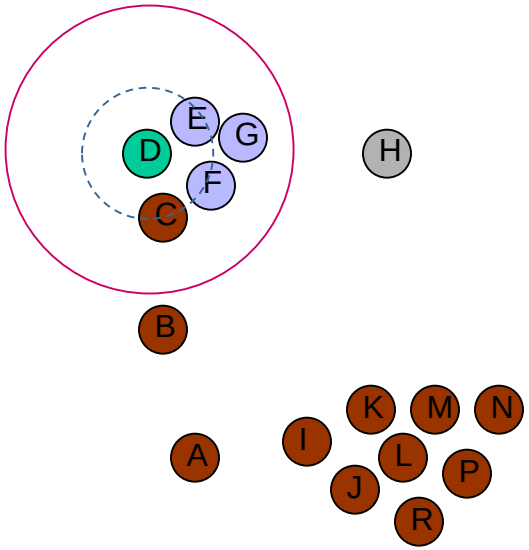


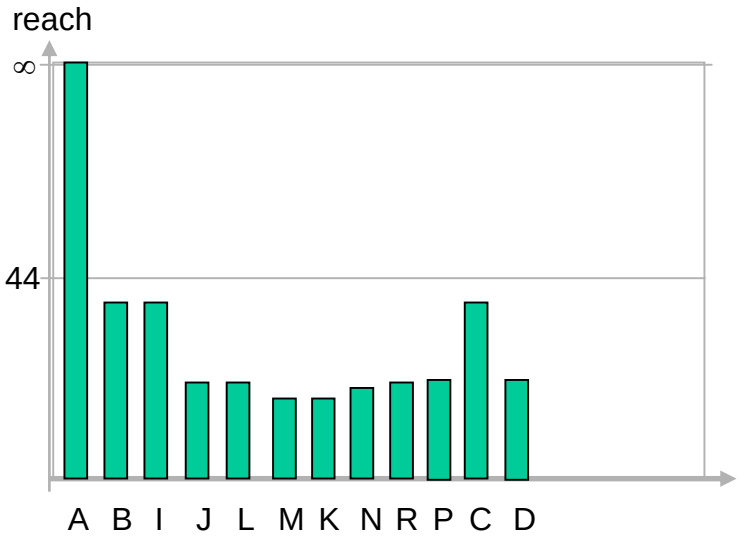


seed list: (D, 22) (F, 22) (E, 30) (G, 35)

The Algorithm OPTICS – Example (13)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

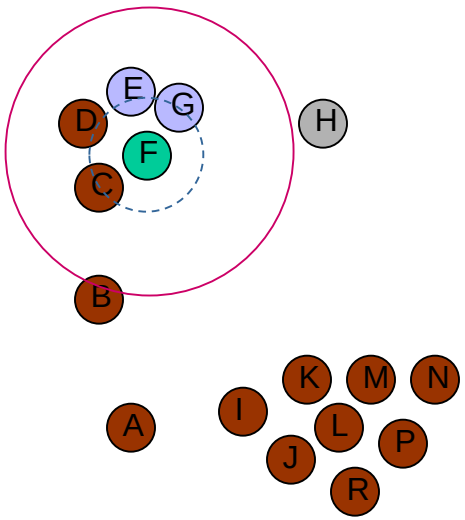


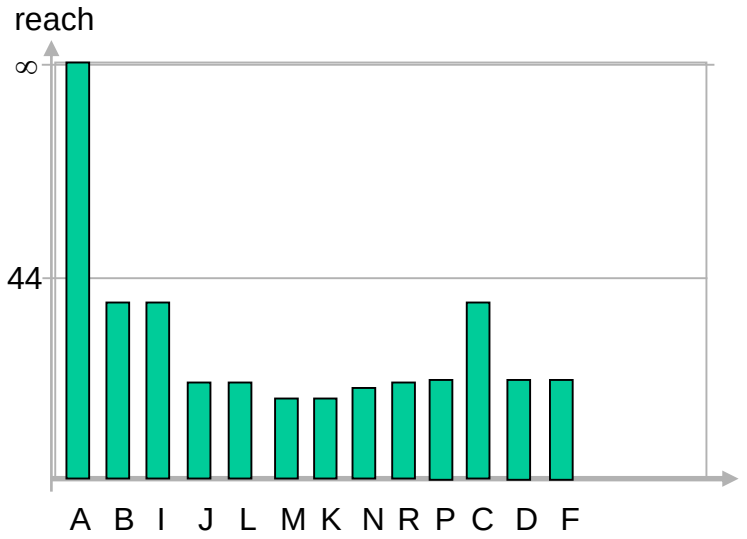


seed list: (F, 22) (E, 22) (G, 32)

The Algorithm OPTICS – Example (14)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

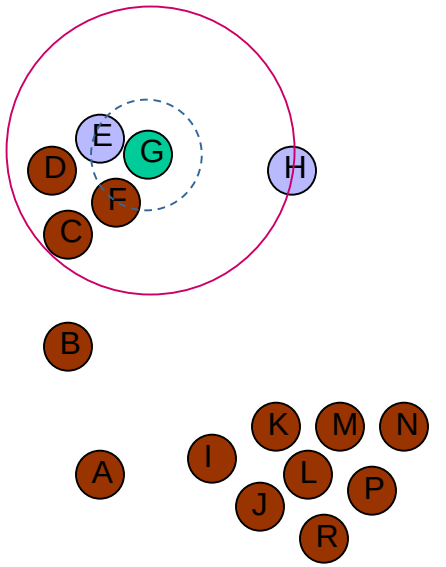


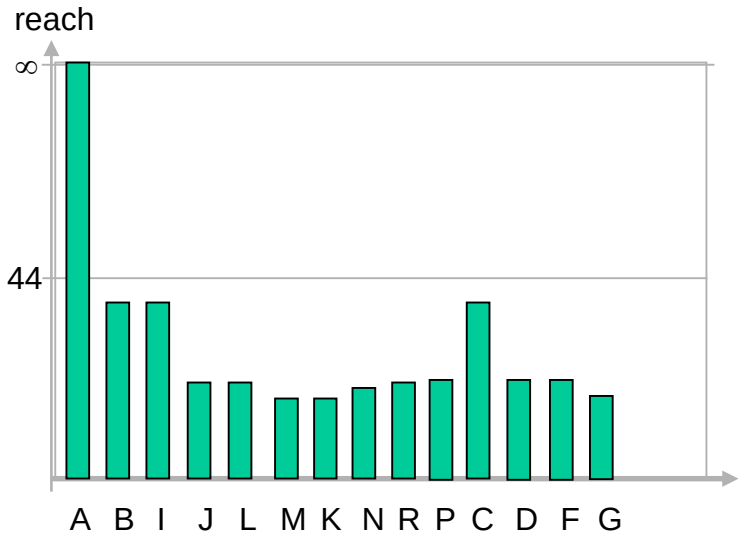


seed list: (G, 17) (E, 22)

The Algorithm OPTICS – Example (15)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

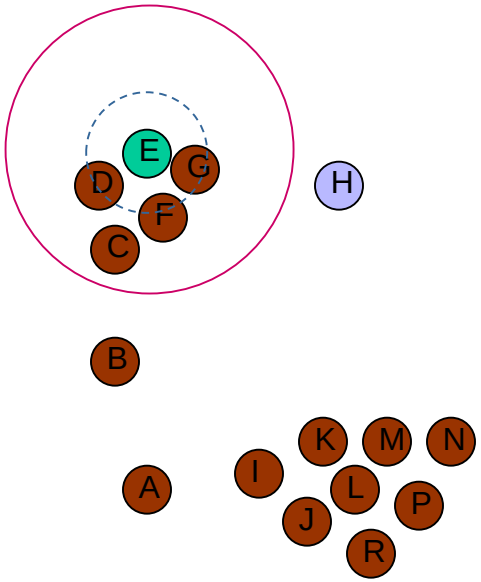


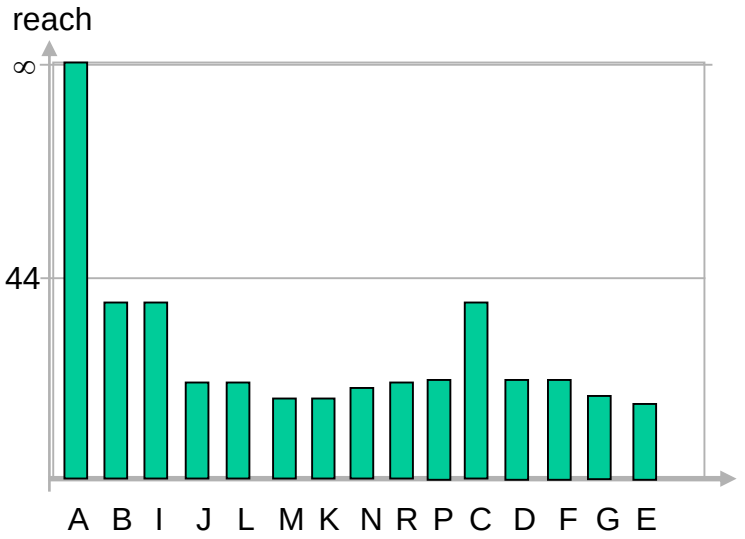


seed list: (E, 15) (H, 43)

The Algorithm OPTICS – Example (16)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

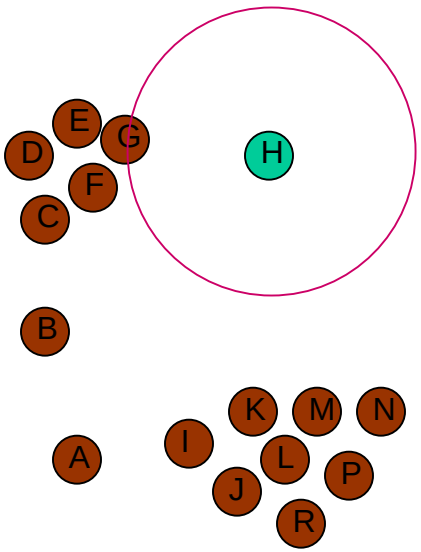


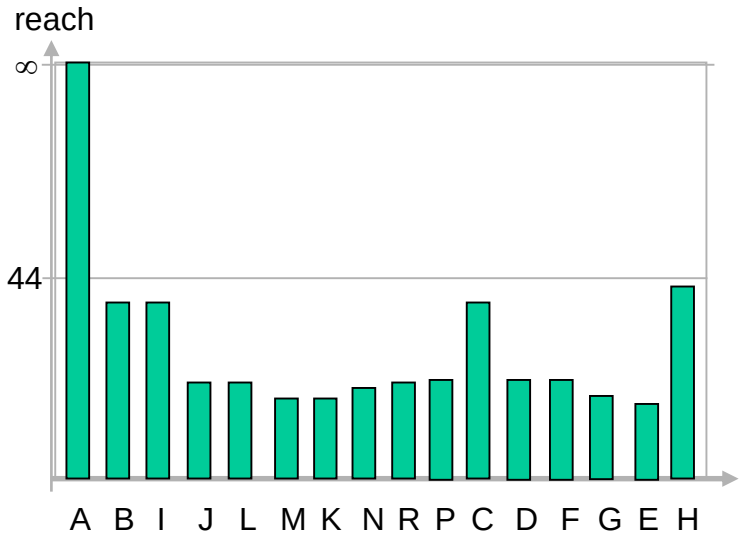


seed list: (H, 43)

The Algorithm OPTICS – Example (17)

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$

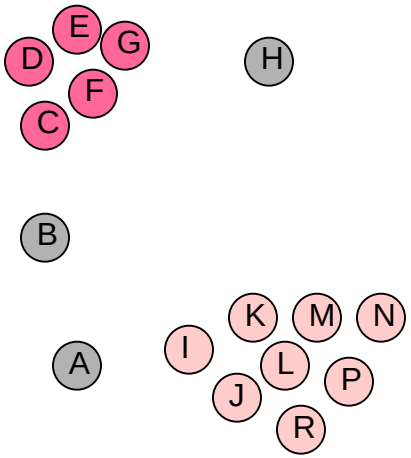


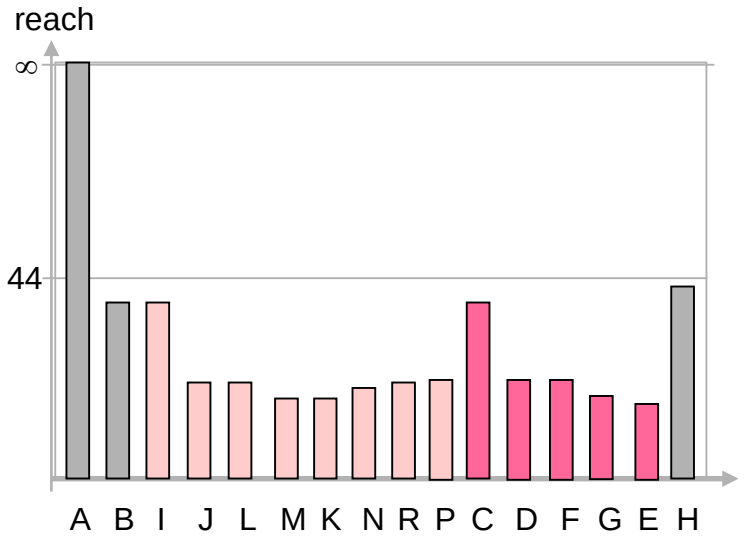


seed list: -

The Algorithm OPTICS – Example (18)

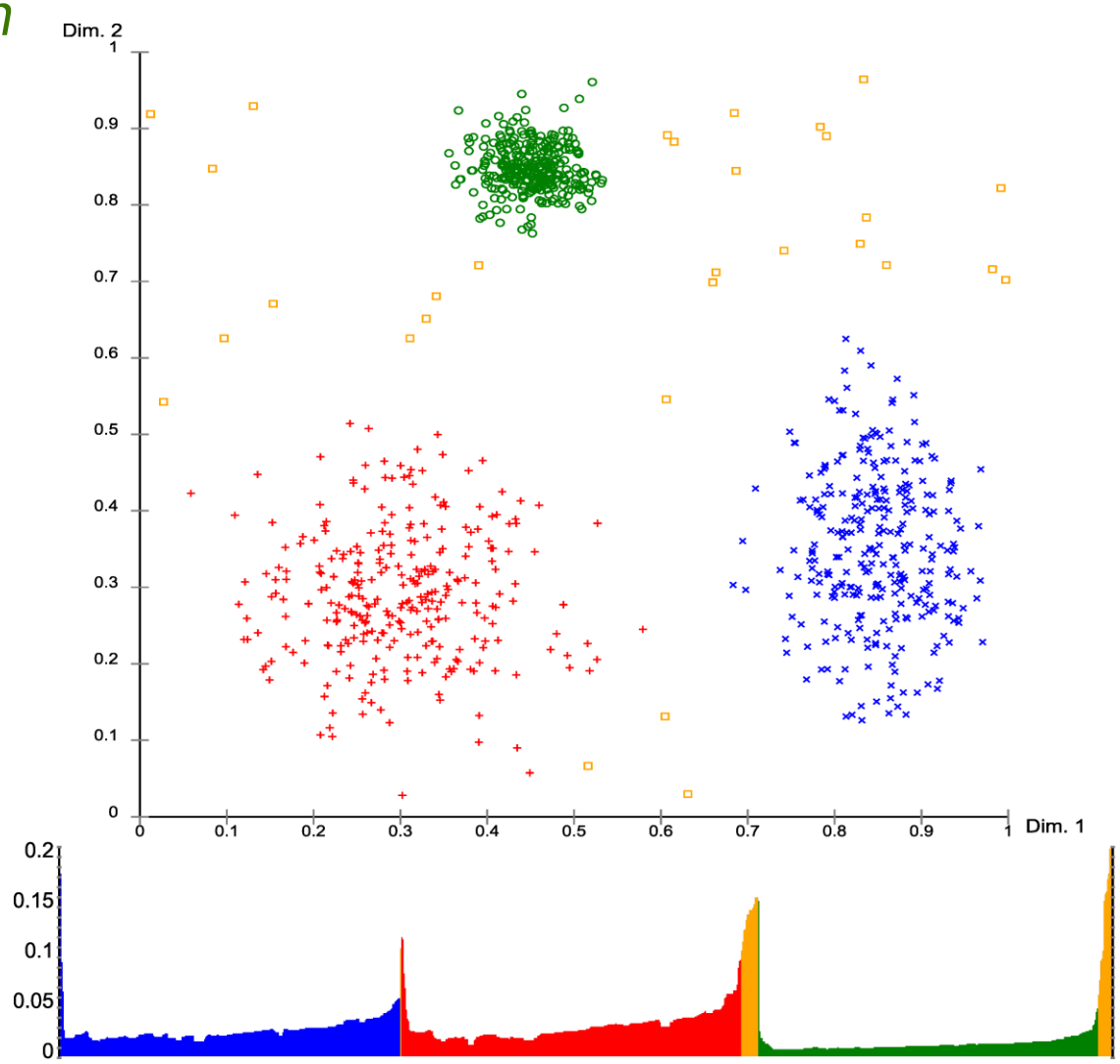
- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$





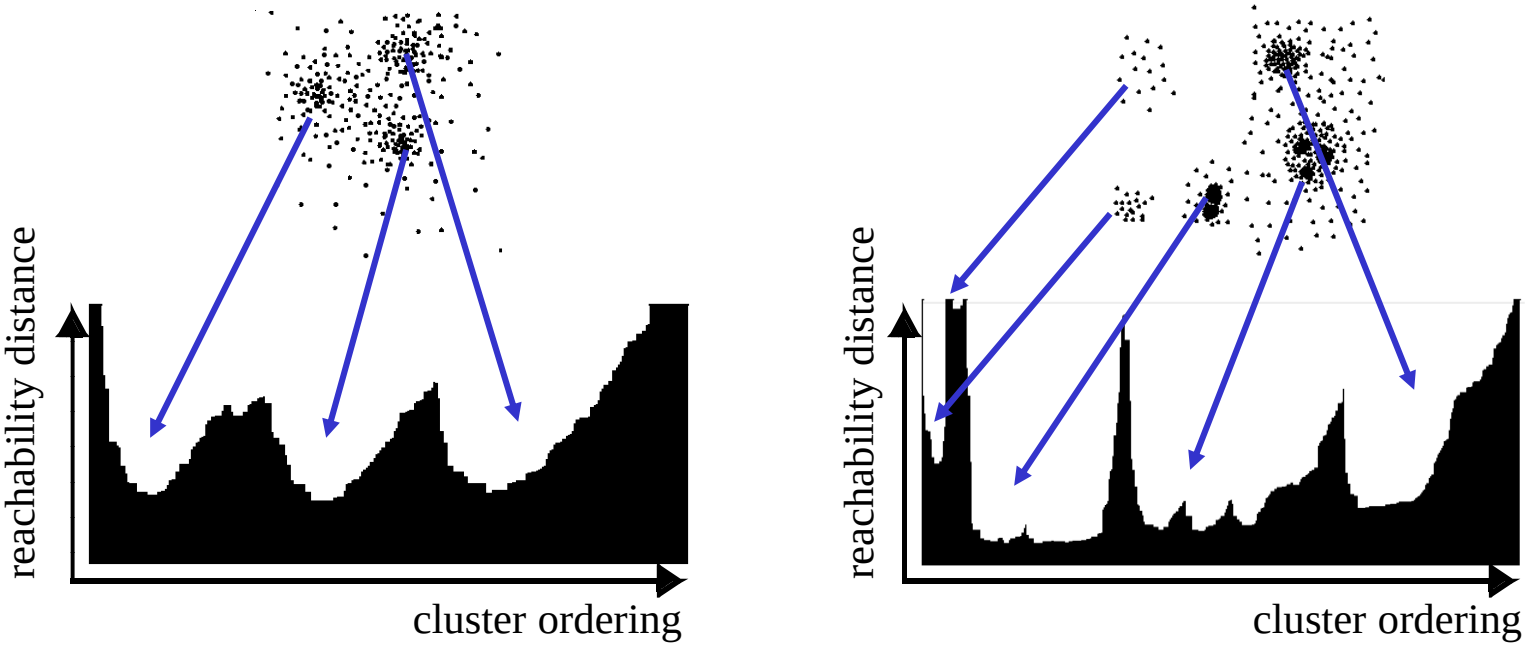
OPTICS: The Reachability Plot

Reachability diagram



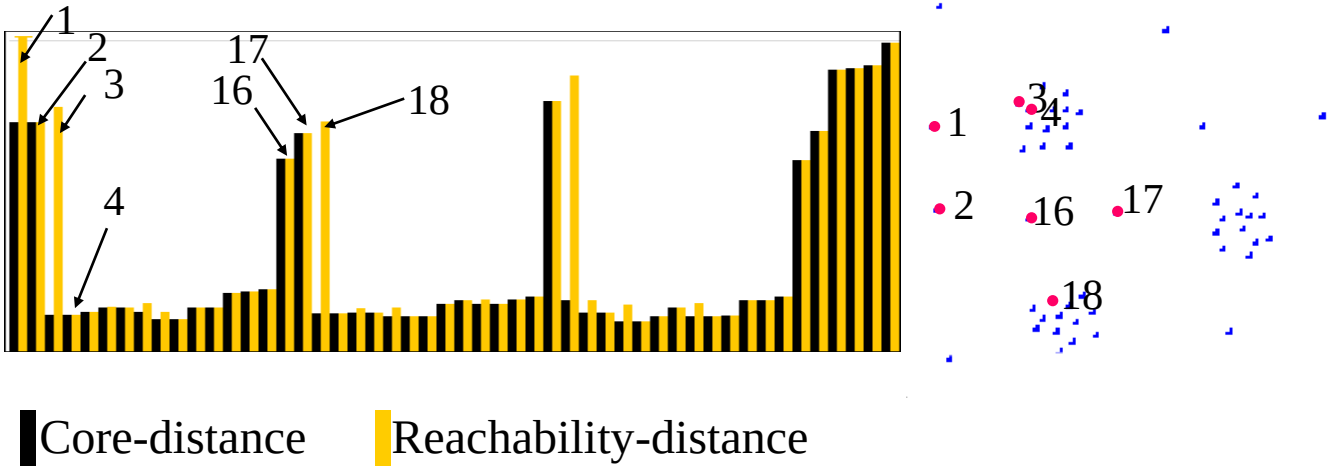
OPTICS: The Reachability Plot

- plot the points together with their reachability-distances. Use the order in which they where returned by the algorithm
  - represents the density-based clustering structure
  - easy to analyze
  - independent of the dimension of the data



OPTICS: Properties

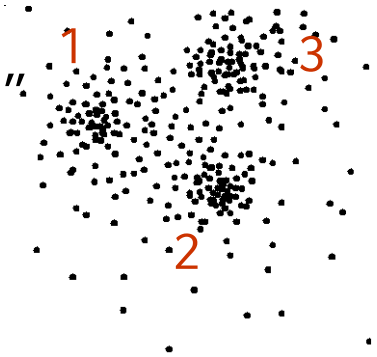
- "Flat" density-based clusters wrt. $\varepsilon^* \leq \varepsilon$ and *MinPts* afterwards:
  - Start with an object *o* where *c-dist*(*o*) $\leq \varepsilon^*$ and *r-dist*(*o*) $> \varepsilon^*$
  - Continue while *r-dist* $\leq \varepsilon^*$



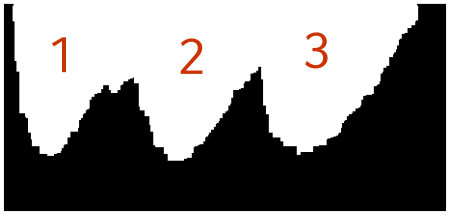
- Performance: approx. runtime(DBSCAN(ε, *MinPts*))
  - O(*n* * runtime(ε-neighborhood-query))
    - without spatial index support (worst case): O(n^2)
    - e.g. tree-based spatial index support: O($n * \log(n)$)

OPTICS: Parameter Sensitivity

- Relatively insensitive to parameter settings
- Good result if parameters are just "large enough"

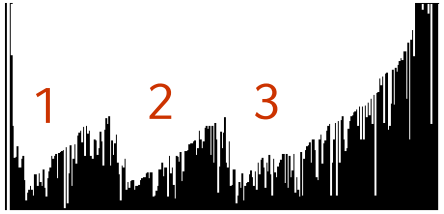


MinPts = 10, ε = 10



MinPts = 10, **ε = 5**

***MinPts* = 2**, ε = 10



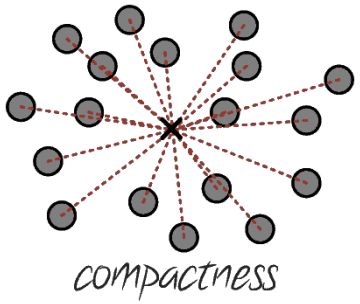
Hierarchical Clustering: Discussion

- Advantages
  - Does not require the number of clusters to be known in advance
  - No (standard methods) or very robust parameters (OPTICS)
  - Computes a complete hierarchy of clusters
  - Good result visualizations integrated into the methods
  - A "flat" partition can be derived afterwards (e.g. via a cut through the dendrogram or the reachability plot)
- Disadvantages
  - May not scale well
    - Runtime for the standard methods: $O(n^2 \log n^2)$
    - Runtime for OPTICS: without index support $O(n^2)$
  - User has to choose the final clustering

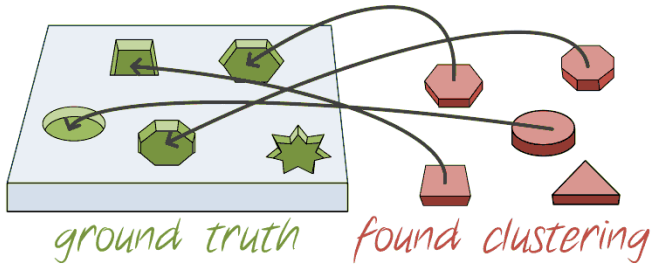
Contents

Evaluation of Clustering Results

- Evaluation based on **expert's** opinion
  - \+ may reveal new insight into the data
  - – very expensive, results are not comparable



- Evaluation based on **internal** measures
  - \+ no additional information needed
  - – approaches optimizing the evaluation criteria will always be preferred



- Evaluation based on **external** measures
  - \+ objective evaluation
  - – needs „ground truth"

  e.g., comparison of two clusterings

Evaluation based on internal measures

Given a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ for Dataset DB

- Sum of square distances: $SSD(\mathcal{C}) = \frac{1}{|DB|} \sum_{C_i \in \mathcal{C}} \sum_{p \in C_i} dist\big(p, \mu(C_i)\big)^2$
- Cohesion: measures the similarity of objects within a cluster
- Separation: measures the dissimilarity of one cluster to another one
- Silhouette Coefficient: combines cohesion and separation

Evaluation based on external measures (1)

Given clustering $\mathcal{C} = (C_1, \ldots, C_k)$ and ground truth $\mathcal{G} = (G_1, \ldots, G_l)$ for dataset DB

- Recall: $rec(C_i, G_j) = \frac{|C_i \cap G_j|}{|G_j|}$ Precision: $prec(C_i, G_j) = \frac{|C_i \cap G_j|}{|C_i|}$
- F-Measure: $F(C_i, G_j) = \frac{2 * rec(C_i, G_j) * prec(C_i, G_j)}{rec(C_i, G_j) + prec(C_i, G_j)}$
- Purity (P): $P(\mathcal{C}, \mathcal{G}) = \sum_{C_i \in \mathcal{C}} \frac{|C_i|}{|DB|} pur(C_i, \mathcal{G})$ $pur(C_i, \mathcal{G}) = \max_{G_j \in \mathcal{G}} prec(C_i, G_j)$
- Rand Index: $RI(\mathcal{C}, \mathcal{G}) = (a + b) / \binom{|DB|}{2}$ „normalized number of agreements “
  $a = \{(o_i, o_j) \in DB \times DB | o_i \neq o_j \wedge \exists C \in \mathcal{C}: o_i, o_j \in C \wedge \exists G \in \mathcal{G}: o_i, o_j \in G\}$
  $b = \{(o_i, o_j) \in DB \times DB | o_i \neq o_j \wedge \neg\exists C \in \mathcal{C}: o_i, o_j \in C \wedge \neg\exists G \in \mathcal{G}: o_i, o_j \in G\}$
- Jaccard Coefficient (JC)
- …

Evaluation based on external measures (2)

Given clustering $\mathcal{C} = (C_1, \ldots, C_k)$ and ground truth $\mathcal{G} = (G_1, \ldots, G_l)$ for dataset DB

- Mutual Entropy: $H(\mathcal{C}|\mathcal{G}) = -\sum_{C_i \in \mathcal{C}} p(C_i) \sum_{G_j \in \mathcal{G}} p(G_j|C_i) log\ p(G_j|C_i)$
  $= -\sum_{C_i \in \mathcal{C}} \frac{|C_i|}{|DB|} \sum_{G_j \in \mathcal{G}} \frac{|C_i \cap G_j|}{|C_i|} * \log_2(\frac{|C_i \cap G_j|}{|C_i|})$

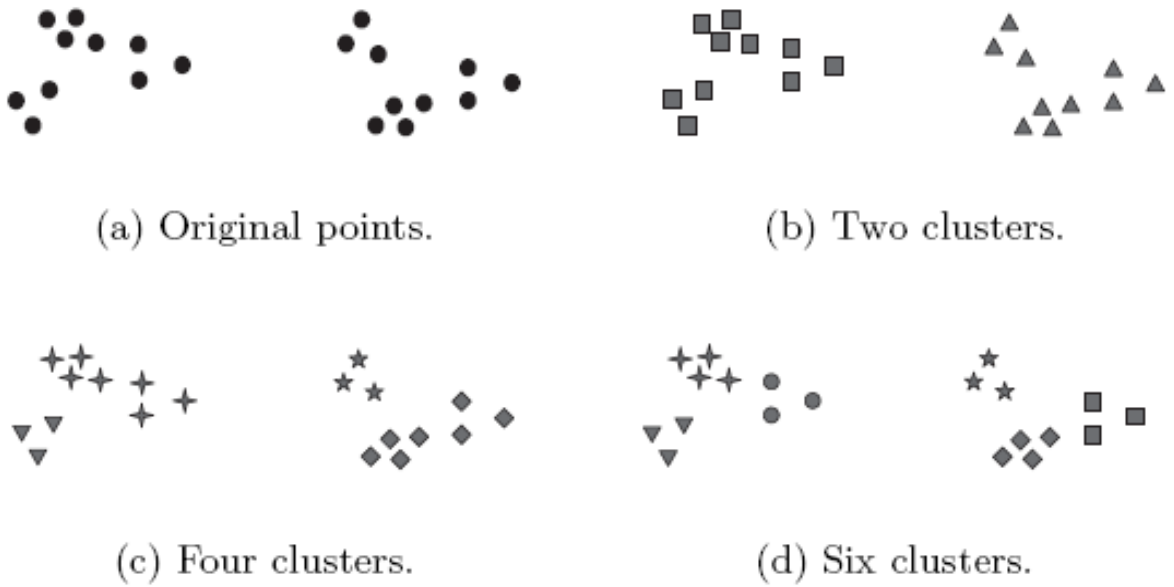
- Mutual Information: $I(\mathcal{C}, \mathcal{G}) = H(\mathcal{C}) - H(\mathcal{C}|\mathcal{G}) = H(\mathcal{G}) - H(\mathcal{G}|\mathcal{C})$

  where entropy $H(\mathcal{C}) = -\sum_{C_i \in \mathcal{C}} p(C_i) \cdot \log p(C_i) = -\sum_{C \in \mathcal{C}} \frac{|C_i|}{|DB|} \cdot \log \frac{|C_i|}{|DB|}$

- Normalized Mutual Information: $NMI(\mathcal{C}, \mathcal{G}) = \frac{I(\mathcal{C},\mathcal{G})}{\sqrt{H(\mathcal{C})H(\mathcal{G})}}$

Ambiguity of Clusterings

Different possibilities to cluster a set of objects



(a) Original points.

(b) Two clusters.

(c) Four clusters.

(d) Six clusters.

from: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

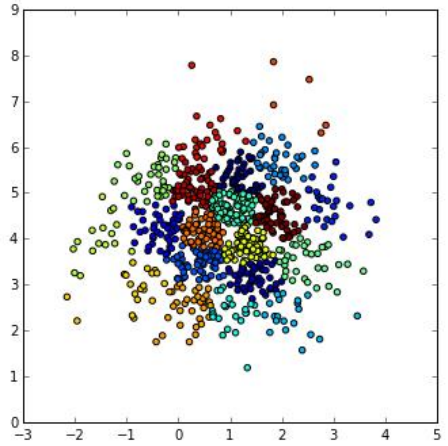
Ambiguity of Clusterings

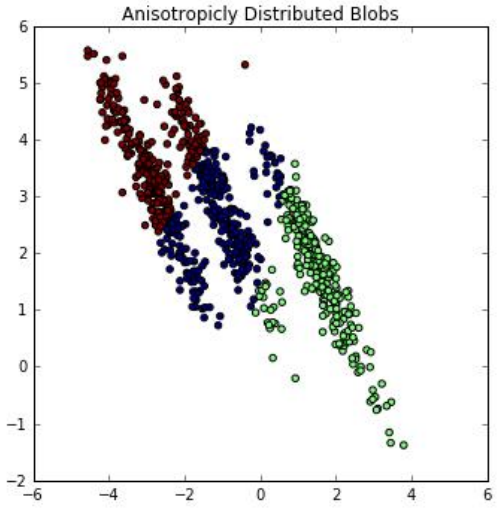
- Kind of a philosophical problem:

  "What is a correct clustering?"
- Most approaches find clusters in every dataset, even in uniformly distributed object

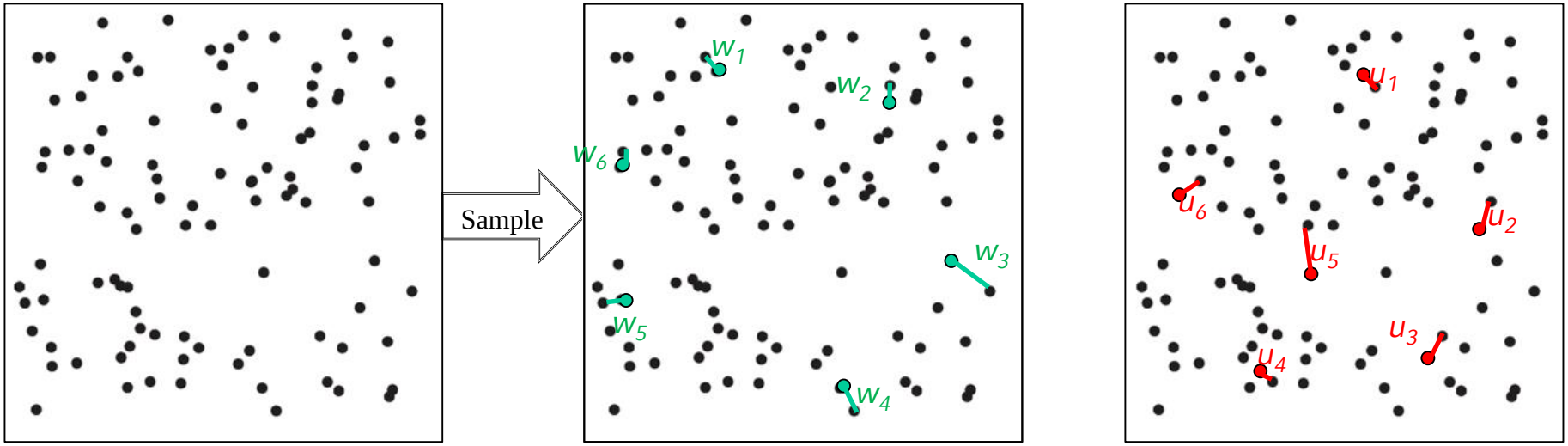
- Are there clusters?
  - Apply clustering algorithm
  - Check for reasonability of clusters
- Problem: No clusters found ⇏ no clusters existing
  - Maybe clusters exists only in certain models, but can not be found by used clustering approach

- Independent of clustering: Is there a data-given cluster tendency?





Hopkins statistics



dataset
(*n* objects)

Random selection
(*m* objects) $m << n$

m uniformly
distributed objects

w_i: distances of selected objects to the next neighbor in dataset

u_i: distances of uniformly distributed objects to the next neighbor in dataset

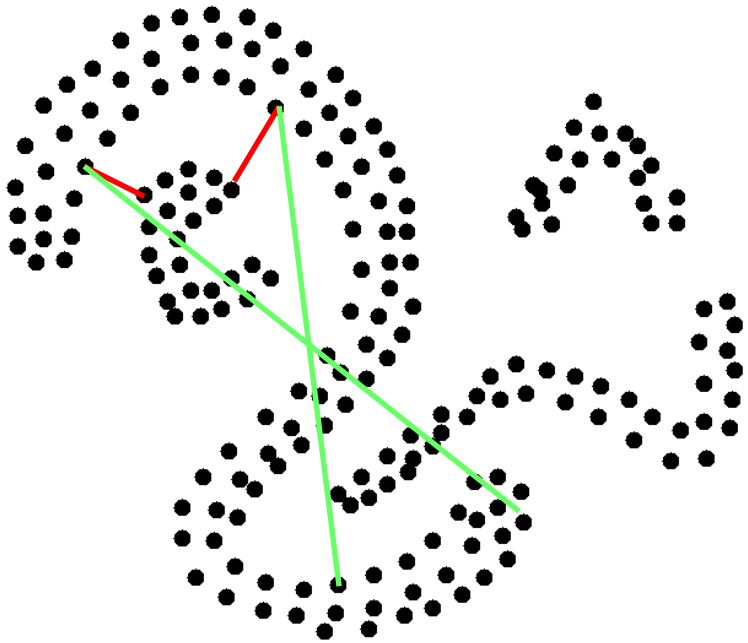
$$H = \frac{\sum_{i=1}^{m} u_i}{\sum_{i=1}^{m} u_i + \sum_{i=1}^{m} w_i}$$

$0 \le H \le 1$

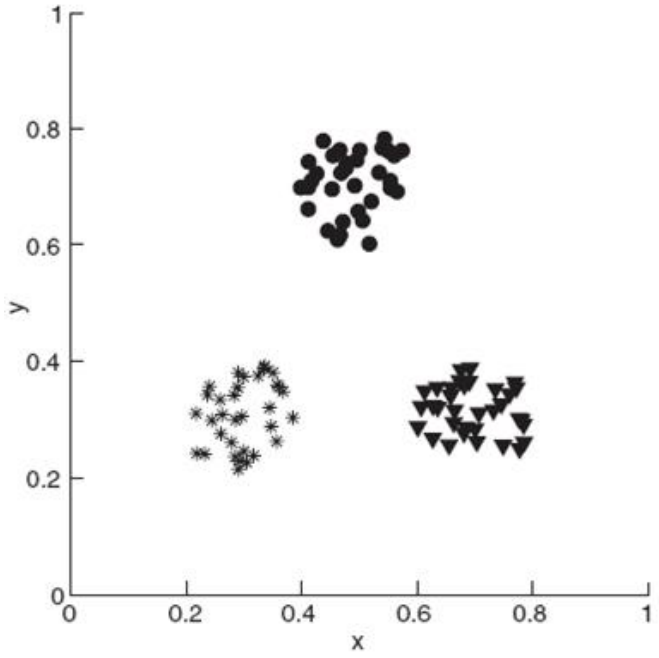
H ≈ 0 : data are very regular (e.g. on grid)
H ≈ 0,5 : data are uniformly distributed
H ≈ 1 : data are strongly clustered

Cohesion and Separation

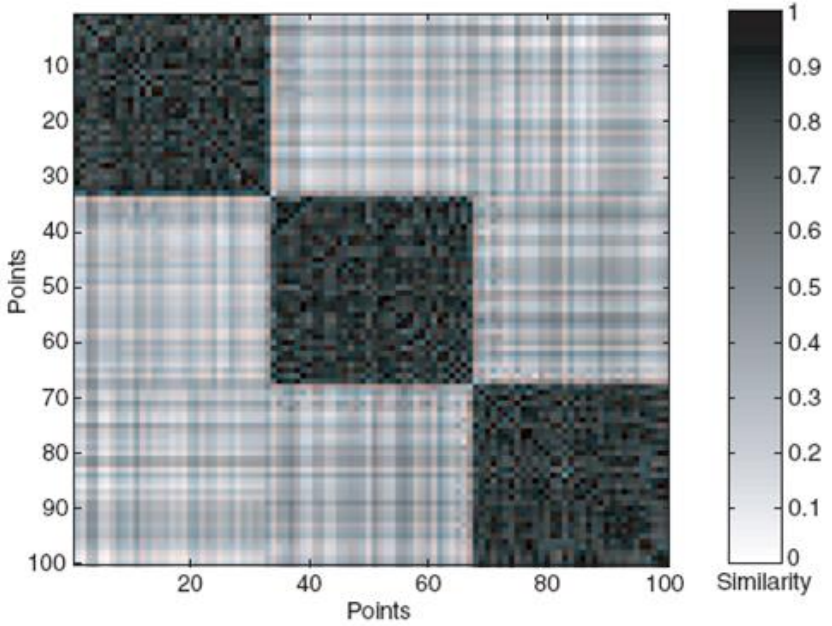
- Suitable for globular cluster, but not for stretched clusters



Evaluating the Similarity Matrix



dataset
(well separated clusters)

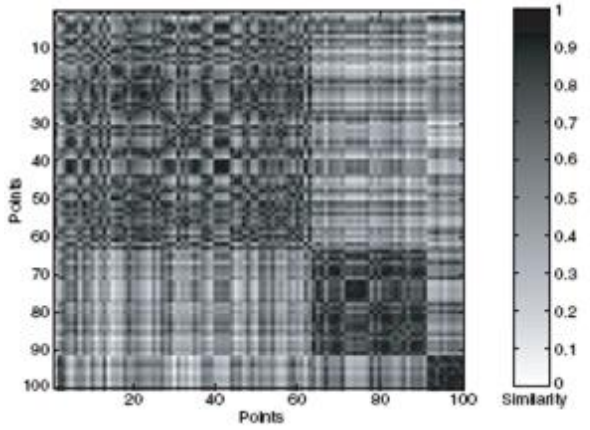


Similarity matrix
(sorted by k-means Cluster-labels)

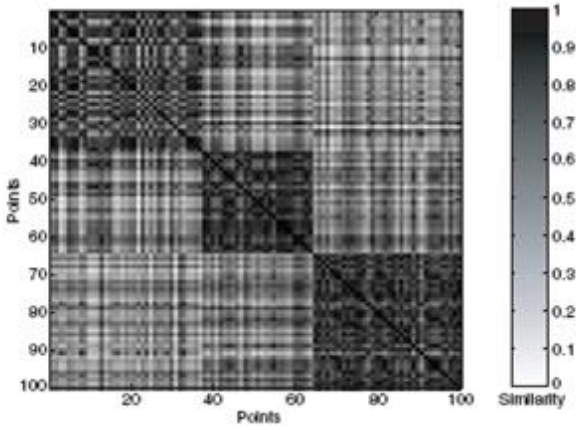
from: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

Evaluating the Similarity Matrix (cont'd)

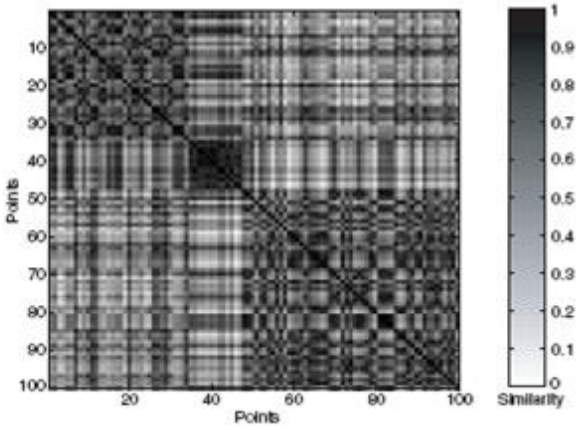
similarity matrices differ for different clustering approaches



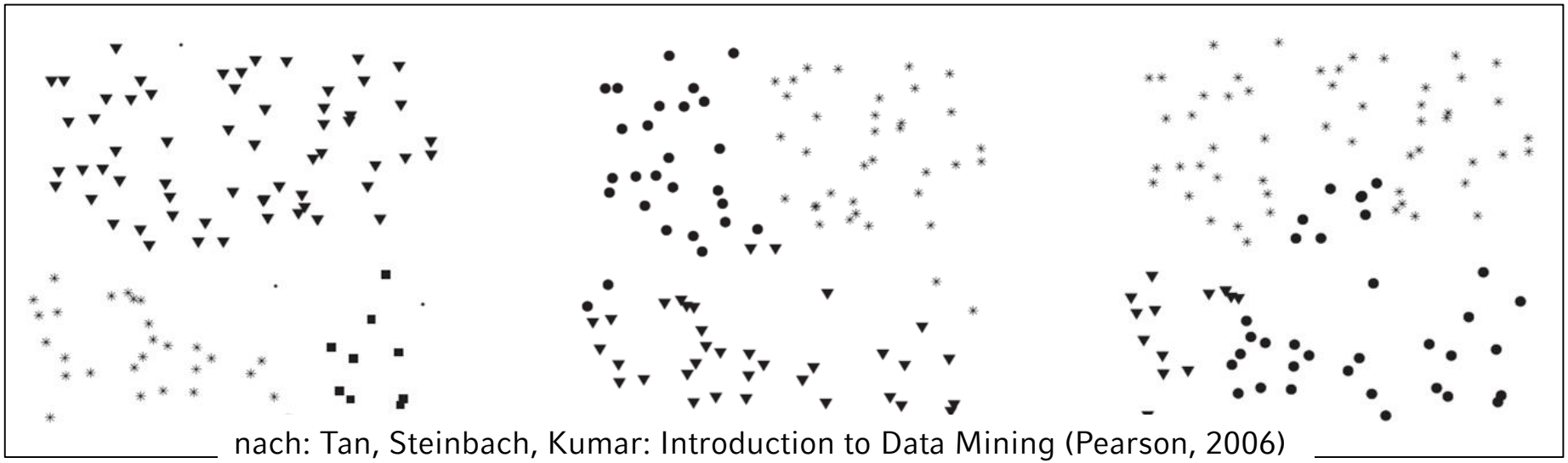
DBSCAN



k-means



complete link

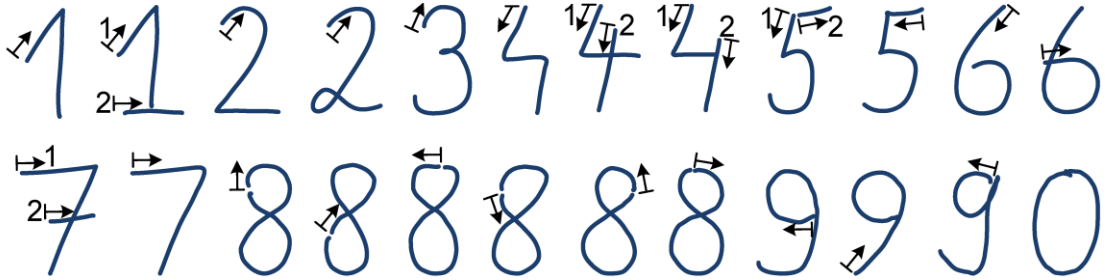


nach: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

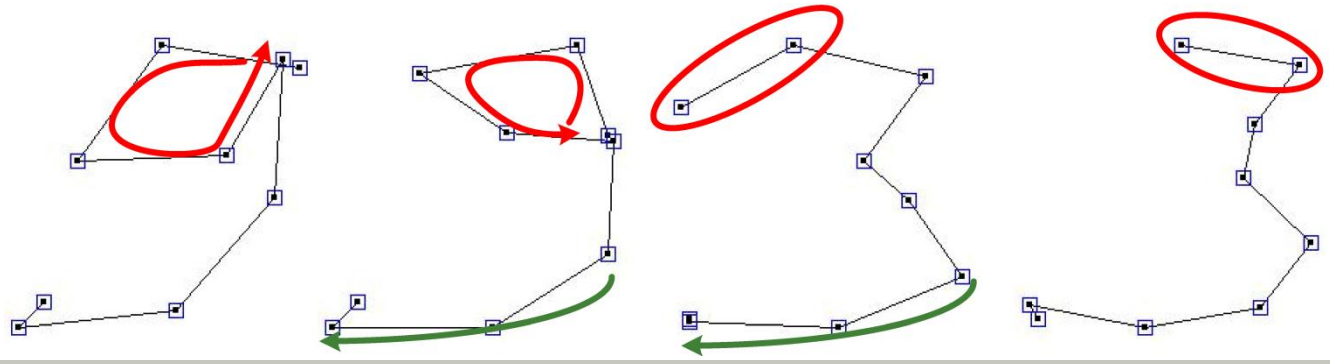
Example: Pen digit clustering

Example Pendigits-dataset:

- Label for each digit: 0,...,9
- Features: temporal ordered points in the draw lines
- Useful concepts depending of the given classes:
  - One digit can have subgroups for varying writing styles



  - Different digits can have similarities



Contents

Ensemble Clustering

Problem:

- Many differing cluster definitions
- Parameter choice usually highly influences the result
- → What is a ‚good' clustering?

Idea: Find a consensus solution (also ensemble clustering) that consolidates multiple clustering solutions.

Benefits of Ensemble Clustering:

- **Knowledge Reuse:** possibility to integrate the knowledge of multiple known, good clusterings
- **Improved quality**: often ensemble clustering leads to "better" results than its individual base solutions.
- **Improved robustness**: combining several clustering approaches with differing data modeling assumptions leads to an increased robustness across a wide range of datasets.
- **Model Selection:** novel approach for determining the final number of clusters
- **Distributed Clustering:** if data is inherently distributed (either feature-wise or object-wise) and each clusterer has only access to a subset of objects and/or features, ensemble methods can be used to compute a unifying result.

Ensemble Clustering – Basic Notions

Given: a set of L clusterings $\mathfrak{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_L\}$ for dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \in \mathbb{R}^D$

Goal : find a consensus clustering $\mathcal{C}^*$

What exactly is a consensus clustering?

We can differentiate between 2 categories for ensemble clustering:

- Approaches based on pairwise similarity
  Idea: find a consensus clustering $\mathcal{C}^*$ for which the similarity function $\phi(\mathfrak{C}, \mathcal{C}^*) = \frac{1}{L}\sum_{l=1}^{L} \phi(\mathcal{C}_l, \mathcal{C}^*)$ is maximal.

  basically our external evaluation measures, which compare two clusterings

- Probabilistic approaches :
  Assume that the L labels for the objects $\mathbf{x}_i \in \mathbf{X}$ follow a certain distribution

→ We will present one exemplary approach for both categories in the following

Ensemble Clustering – Similarity-based Approaches

Given: a set of L clusterings $\mathfrak{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_L\}$ for dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \in \mathbb{R}^D$

- Goal : find a consensus clustering $\mathcal{C}^*$ for which the similarity function $\phi(\mathfrak{C}, \mathcal{C}^*) = \frac{1}{L}\sum_{l=1}^{L} \phi(\mathcal{C}_l, \mathcal{C}^*)$ is maximal.
  - Popular choices for ϕ in the literature:
    - **Pair counting-based measures:** Rand Index (RI), Adjusted RI, Probabilistic RI
    - **Information theoretic measures:** Mutual Information (I), Normalized Mutual Information (NMI), Variation of Information (VI)

Problem: the above objective is intractable

Solutions:

- Methods based on the co-association matrix (related to RI)
- Methods using cluster labels without co-association matrix (often related to NMI)
  - Mostly graph partitioning
  - Cumulative voting

Ensemble Clust. – Approaches based on Co-Association

Given: a set of L clusterings $\mathfrak{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_L\}$ for dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \in \mathbb{R}^D$

The co-association matrix $\mathbf{S}^{(\mathfrak{C})}$ is an $N \times N$ matrix representing the label similarity of object pairs: $s_{i,j}^{(\mathfrak{C})} = \sum_{l=1}^{L} \delta\left(C(\mathcal{C}_l, \mathbf{x}_i), C(\mathcal{C}_l, \mathbf{x}_j)\right)$

where $\delta(a,b) = \begin{cases} 1 & if\ a = b \\ 0 & else \end{cases}$

cluster label of $\mathbf{x}_i$, $\mathbf{x}_j$ in clustering $\mathcal{C}_l$

Based on the similarity matrix defined by $\mathbf{S}^{(\mathfrak{C})}$ traditional clustering approaches can be used

Often $\mathbf{S}^{(\mathfrak{C})}$ is interpreted as weighted adjacency matrix, such that methods for graph partitioning can be applied.

In [Mirkin'96] a connection of consensus clustering based on the co-association matrix and the optimization of the pairwise similarity based on the Rand Index ($\mathcal{C}^{best} = argmax_{\mathcal{C}^*} \left\{\frac{1}{L}\sum_{\mathcal{C}_l \in \mathfrak{C}} RI(\mathcal{C}_l, \mathcal{C}^*)\right\}$) has been proven.

[Mirkin'96] B. Mirkin: *Mathematical Classification and Clustering*. Kluwer, 1996.

Ensemble Clust. – Approaches based on Cluster Labels

- Consensus clustering $\mathcal{C}^*$ for which $\frac{1}{L}\sum_{\mathcal{C}_l \in \mathbb{C}} \phi(\mathcal{C}_l, \mathcal{C}^*)$ is maximal

→ Information theoretic approach: choose ϕ as mutual information (I), normalized mutual information (NMI), information bottleneck (IB),…

- Problem: Usually a hard optimization problem

→ Solution 1: Use meaningful optimization approaches (e.g. gradient descent) or heuristics to approximate the best clustering solution (e.g. [SG02])

→ Solution 2: Use a similar but solvable objective (e.g. [TJP03])

- Idea: use as objective $\mathcal{C}_{best} = argmax_{\mathcal{C}^*}\left\{\frac{1}{L}\sum_{\mathcal{C}_l \in \mathbb{C}} I^s(\mathcal{C}_l, \mathcal{C}^*)\right\}$
  where I^s is the mutual information based on the generalized entropy of degree s: $H^s(X) = (2^{1-s}-1)^{-1}\sum_{x_i \in X}(p_i^s - 1)$

→ For $s = 2$, $I^s(\mathcal{C}_l, \mathcal{C}^*)$ is equal to the category utility function whose maximization is proven to be equivalent to the minimization of the square-error clustering criterion

→ Thus apply a simple label transformation and use e.g. K-Means

[SG02] A. Strehl, J. Ghosh: *Cluster ensembles - a knowledge reuse framework for combining multiple partitions*. Journal of Machine Learning Research, 3, 2002, pp. 583-617.
[TJP03]A. Topchy, A.K. Jain, W. Punch. *Combining multiple weak clusterings*. In ICDM, pages 331-339, 2003.

Ensemble Clustering – A Probabilistic Approach

Assumption 1: all clusterings $\mathcal{C}_l \in \mathfrak{C}$ are partitionings of the dataset $\mathbf{X}$.

Assumption 2: there are K^* consensus clusters

The dataset $\mathbf{X}$ is represented by the set

cluster label of x_n in clustering $\mathcal{C}_l$

$$\mathbf{Y} = \{\boldsymbol{y}_n \in \mathbb{N}_0^L \mid \exists \mathbf{x}_n \in \mathbf{X}. \forall \mathcal{C}_l \in \mathfrak{C}. y_{nl} = C(\mathcal{C}_l, \mathbf{x}_n)\}$$

we have a new feature Space $\mathbb{N}_0^L$, where the l^{th} feature represents the cluster labels from partition $\mathcal{C}_l$

Assumption 3: the dataset $\mathbf{Y}$ (labels of base clusterings) follow a multivariate mixture distribution:

conditional independence assumptions for $\mathcal{C}_l \in \mathfrak{C}$

$$P(\mathbf{Y}|\boldsymbol{\Theta}) = \prod_{n=1}^{N} \sum_{k=1}^{K^*} \alpha_k P_k(\boldsymbol{y}_n|\boldsymbol{\theta}_k) = \prod_{n=1}^{N} \sum_{k=1}^{K^*} \alpha_k \prod_{l=1}^{L} P_{kl}(y_{nl}|\boldsymbol{\theta}_{kl})$$

$P_{kl}(y_{nl}|\boldsymbol{\theta}_{kl}) \sim M\left(1, \left(p_{k,l,1}, \ldots, p_{k,l,|\mathcal{C}_l|}\right)\right)$ follows a $|\mathcal{C}_l|$-dimensional multinomial distribution: $P_{kl}(y_{nl}|\boldsymbol{\theta}_{kl}) = \prod_{k'=1}^{|\mathcal{C}_l|} p_{k,l,k'}^{\delta(y_{nl},k')}$

therefore: $\boldsymbol{\theta}_{kl} = (p_{k,l,1}, \ldots, p_{k,l,|\mathcal{C}_l|})$

Goal: find the parameters $\boldsymbol{\Theta} = (\alpha_1, \boldsymbol{\theta}_1, \ldots, \alpha_{K^*}, \boldsymbol{\theta}_{K^*})$ such that the likelihood $P(\mathbf{Y}|\boldsymbol{\Theta})$ is maximized

Solution: optimizing the parameters via the EM approach. *(details omitted)*

Presented approach: Topchy, Jain, Punch: *A mixture model for clustering ensembles.* In ICDM, pp. 379-390, 2004.
Later extensions: H. Wang, H. Shan, A. Banerjee: *Bayesian cluster ensembles.* In ICDM, pp. 211-222, 2009.
P. Wang, C. Domeniconi, K. Laskey: *Nonparametric Bayesian clustering ensembles.* In PKDD, pp. 435-450, 2010.

Contents

1) Introduction to clustering
2) Partitioning Methods
   - K-Means
   - K-Medoid
   - Choice of parameters: Initialization, Silhouette coefficient
3) Expectation Maximization: a statistical approach
4) Density-based Methods: DBSCAN
5) Hierarchical Methods
   - Agglomerative and Divisive Hierarchical Clustering
   - Density-based hierarchical clustering: OPTICS
6) Evaluation of Clustering Results
7) Further Clustering Topics
   - Ensemble Clustering
   - Discussion: an alternative view on DBSCAN

Reconsider DBSCAN algorithm

- Standard DBSCAN evaluation is based on recursive database traversal.
- Böhm et al. (2000) observed that DBSCAN, among other clustering algorithms, may be efficiently built on top of similarity join operations.

Similarity joins

- An ε-similarity join yields all pairs of ε-similar objects from two data sets P, Q:

$$P \bowtie_\varepsilon Q = \{(p,q) | \, p \in P \wedge q \in Q \wedge dist(p,q) \leq \varepsilon\}$$

  - SQL-Query: SELECT * FROM P, Q WHERE $dist(\mathrm{P},\mathrm{Q}) \leq \varepsilon$

- An ε-similarity self join yields all pairs of ε-similar objects from a database DB:

$$DB \bowtie_\varepsilon DB = \{(p,q) | \, p \in DB \wedge q \in DB \wedge dist(p,q) \leq \varepsilon\}$$

  - SQL-Query: SELECT * FROM DB p, DB q WHERE $dist(p,q) \leq \varepsilon$

Böhm C., Braunmüller, B., Breunig M., Kriegel H.-P.: High performance clustering based on the similarity join. CIKM 2000: 298-305.

Similarity Join for Density-Based Clustering

ε-Similarity self join:

$$DB \bowtie_\varepsilon DB = \{(p,q) | \, p \in DB \wedge q \in DB \wedge dist(p,q) \leq \varepsilon\}$$

Relation „directly ε, *MinPts*-density reachable“ may be expressed in terms of an ε-similarity self join:

$$ddr_{\varepsilon,\mu} = \{(p,q) | \, p \text{ is } \varepsilon,\mu\text{–core point} \wedge q \in N_\varepsilon(p)\}$$

$$= \{(p,q) | p,q \in DB \wedge dist(p,q) \leq \varepsilon \wedge \exists_{\geq \mu} q' \in DB : dist(p,q') \leq \varepsilon\}$$

$$= \{(p,q) | (p,q) \in DB \bowtie_\varepsilon DB \wedge \exists_{\geq \mu} q' : (p,q') \in DB \bowtie_\varepsilon DB\}$$

$$= \sigma_{|\pi_p(DB \bowtie_\varepsilon DB)| \geq \mu}(DB \bowtie_\varepsilon DB) =: DB \bowtie_{\varepsilon,\mu} DB$$

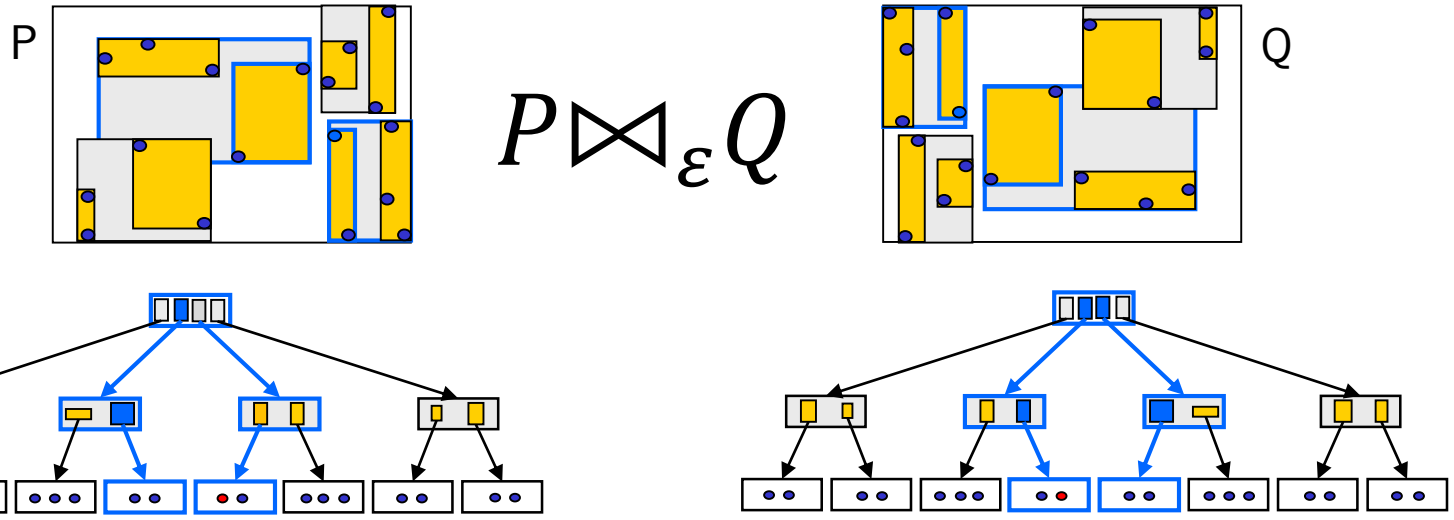
- SQL-Query: SELECT * FROM DB p, DB q WHERE $dist(p,q) \leq \varepsilon$ GROUP BY p.id HAVING $\text{count}(p.id) \geq \mu$
- Remark: $DB \bowtie_\varepsilon DB$ is a symmetric relation, $ddr_{\varepsilon,\mu} = DB \bowtie_{\varepsilon,\mu} DB$ is not.

DBSCAN then computes the connected components within $DB \bowtie_{\varepsilon,\mu} DB$.

Efficient Similarity Join Processing

For very large databases, efficient join techniques are available

- Block nested loop or index-based nested loop joins exploit secondary storage structure of large databases.
- Dedicated similarity join, distance join, or spatial join methods based on spatial indexing structures (e.g., R-Tree) apply particularly well. They may traverse their hierarchical directories in parallel (see illustration below).
- Other join techniques including sort-merge join or hash join are not applicable.



Clustering – Summary

Partitioning Methods: K-Means, K-Medoid, K-Mode, K-Median

Probabilistic Model-Based Clusters: Expectation Maximization

Density-based Methods: DBSCAN

Hierarchical Methods

- Agglomerative and Divisive Hierarchical Clustering
- Density-based hierarchical clustering: OPTICS

Evaluation of Clustering Results

- Evaluation based on an expert's knowledge; internal evaluation measures; external evaluation measures

Further Clustering Topics

- Ensemble Clustering: finding a consensus clustering agreeing with multiple base clustering and its advantages
  - co-assocaition matrix, information theoretic approaches, probabilistic approaches
- Discussion of DBSCAN as Join operation