Cluster Analysis

Contents of this Chapter

Introduction

Representative-based clustering [Aggarwal 2015, section 6.3]

Probabilistic model-based clustering [Section 6.5]

Hierarchical clustering [Section 6.4]

Density-based clustering [Section 6.6]

Non-negative matrix factorization [Section 6.8]

Cluster ensembles [Section 7.7]

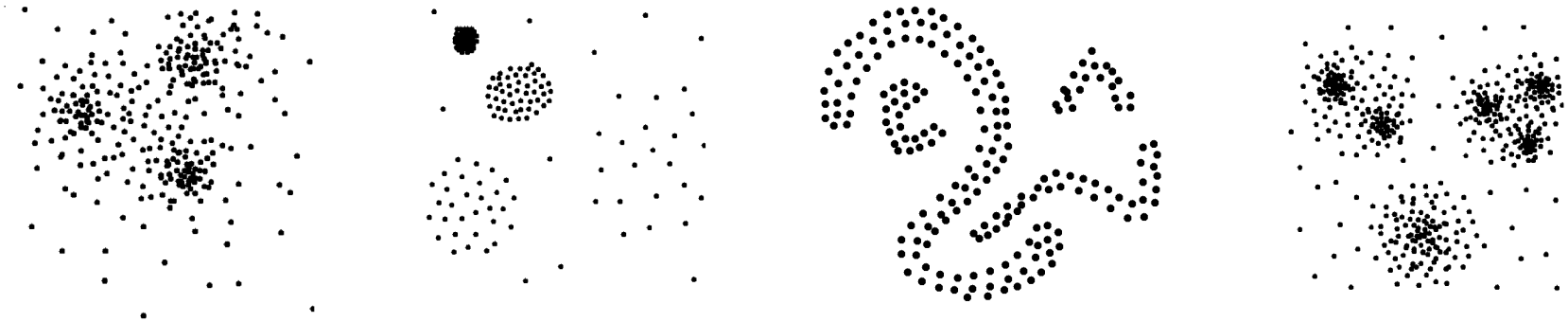
High-dimensional clustering [Section 7.4]

Semi-supervised clustering [Section 7.5]

Introduction

Goal of Cluster Analysis

- Identification of a finite set of categories, classes or groups (*clusters*) in the dataset.
- Objects within the *same* cluster shall be as similar as possible.
- Objects of *different* clusters shall be as dissimilar as possible.



Clusters of different sizes, shapes, densities

Hierarchical clusters

Disjoint / overlapping clusters

Introduction

Clustering as Optimization Problem

Definition

- Dataset D, $|D| = n$
- Objects $o \in O$ have d *attributes* (also called *features/dimensions*)
- *Clustering C of D*: $C = \{C_1, \ldots, C_k\}$

  where $C_i \subseteq D$ and $\bigcup_{i,1 \le i \le k} C_i = D$

Goal

Find *clustering* that „best fits“ the given dataset.
Degree of fit measured by a *score* function.

Search space

Space of all clusterings
Size is $O(2^n)$ for $k = 2$ clusters

Local optimization methods

Introduction

Clustering as Optimization Problem

Steps

1. Choice of model category
   partitioning, hierarchical, density-based, etc.
2. Definition of score function
   typically, based on distance function
3. Choice of model structure
   feature selection / number of clusters
4. Search for model parameters
   clusters / cluster representatives

Distance Functions

Basics

Formalizing similarity

- Sometimes: similarity function
- Typically: distance function $dist(o_1,o_2)$ for pairs of objects o_1 and o_2
- Small distance ≈ similar objects
- Large distance ≈ dissimilar objects

Requirements for distance functions

(1) $dist(o_1, o_2) = d \in \mathbb{R}^{\geq 0}$

(2) $dist(o_1, o_2) = 0$ iff $o_1 = o_2$

(3) $dist(o_1, o_2) = dist(o_2, o_1)$ (symmetry)

(4) Additionally for metric distance functions (triangle inequality)

$$dist(o_1, o_3) \leq dist(o_1, o_2) + dist(o_2, o_3).$$

Distance Functions

Distance Functions for Numerical Attributes

Objects $x = (x_1, ..., x_d)$ and $y = (y_1, ..., y_d)$

L_p-Metric (Minkowski-Distance) $$dist(x,y) = \sqrt[p]{\sum_{i=1}^{d} (x_i - y_i)^p}$$

Euclidean Distance ($p = 2$) $$dist(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

Manhattan-Distance ($p = 1$) $$dist(x,y) = \sum_{i=1}^{d} |x_i - y_i|$$

Maximum-Metric ($p = \infty$) $$dist(x,y) = \max\{|x_i - y_i| | 1 \le i \le d\}$$

→ A popular *similarity function*: *Correlation Coefficient* $\in$ [-1,+1]

Distance Functions

Other Distance Functions

- For categorical attributes

  Hamming distance $dist(x,y) = \sum_{i=1}^{d} \delta(x_i, y_i)$ *where* $\delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{else} \end{cases}$

- For text documents D (in vector space representation)

  cosine similarity

  $\text{cossim}(x,y) = \frac{< x, y >}{|x| \cdot |y|}$ *with* $< .,. >$ *dot product and* $|.|$ *length of the vector*

$\text{cosdist}(x,y) = 1 - \text{cossim}(x,y)$ corresponding *distance* function

Adequate distance function is crucial for the clustering quality!

Typical Clustering Applications

Overview

Market segmentation

- Clustering the set of customer transactions

Determining user groups on the WWW

- Clustering web-logs

Structuring large sets of text documents

- Hierarchical clustering of the text documents

Generating thematic maps from satellite images

- Clustering sets of raster images of the same area (feature vectors)

Typical Clustering Applications

Determining User Groups on the WWW

Entries of a Web-Log

```
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:44:50 +0100] "GET /~lopa/ HTTP/1.0" 200 1364
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:45:11 +0100] "GET /~lopa/x/ HTTP/1.0" 200 712
fixer.sega.co.jp unknown - [04/Mar/1997:01:58:49 +0100] "GET /dbs/porada.html HTTP/1.0" 200 1229
scooter.pa-x.dec.com unknown - [04/Mar/1997:02:08:23 +0100] "GET /dbs/kriegel_e.html HTTP/1.0" 200 1241
```

Sessions

Session::= <IP-Adress, User-Id, [URL_1, . . ., URL_k]>

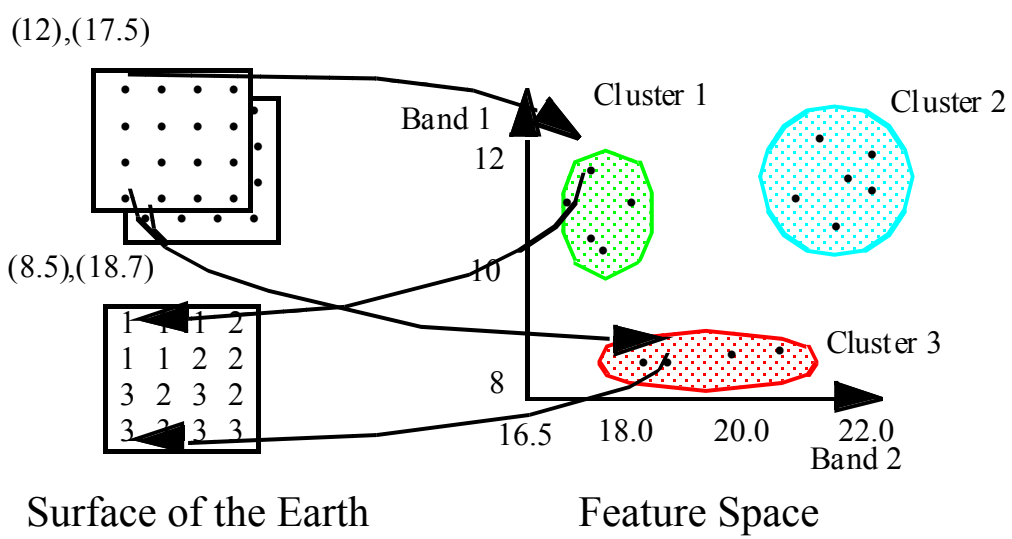
⇒ Which entries form a session?

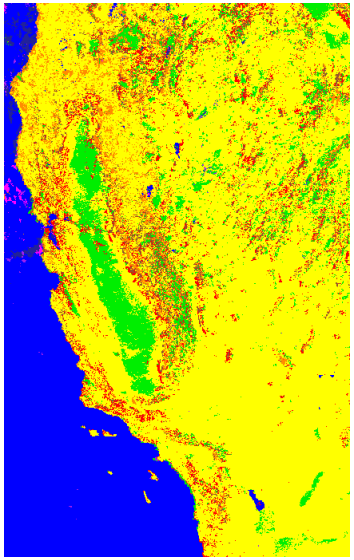
Similarity function for sets of objects (sessions)

$$d(x,y) = \frac{|x \cap y|}{|x \cup y|} \quad \textit{Jaccard Coefficient}$$

Typical Clustering Applications

Generating Thematic Maps from Satellite Images





Assumption

Different land usages exhibit different, characteristic properties of reflection and emission.

Types of Clustering Methods

Representative-based (Partitioning) Clustering

- Parameters: number *k* of clusters, distance function.
- Determines a „flat“ clustering into *k* clusters (with minimal costs).

Probabilistic Model-Based Clustering

- Parameters: number *k* of clusters.
- Determines a „flat“ clustering into *k* clusters (with maximum data likelihood).

Hierarchical Clustering

- Parameters: distance function for objects and for clusters.
- Determines a hierarchy of clusterings, merges always the most similar clusters.

Density-Based Clustering

- Parameters: minimum density within a cluster, distance function.
- Extends cluster by neighboring objects as long as the density is large enough.

Cluster Validation

How good is a given clustering?

Which of several given clusterings is better?

Internal validation criteria

- Measure the compactness of clusters and their separation from each other.
- Often the objective function of some algorithm.
- Hard to compare clusterings generated by different algorithms.

External validation criteria

- Compare discovered clustering to some ground truth clustering (class labels).
- Avoids the bias of internal validation criteria.
- But class labels often unavailable.
- Class labels may not correspond to “natural” clusters.

Cluster Validation

Internal Validation Criteria

Sum of square distances to representatives

- Compute distance to the nearest cluster representative.

Intracluster to intercluster distance ratio

- Sample *r* pairs of data objects. Let *P* be the set of pairs that belong to the same cluster, and let *Q* be the set of the remaining pairs.
- Compute the ratio of the average distance of the pairs in *P* and the average distance of the pairs in *Q*.

Silhouette coefficient

- Compares distances to nearest and second nearest cluster representative.

Data likelihood

- For probabilistic model-based clustering methods.

Cluster Validation

External Validation Criteria

Confusion matrix

- Rows: classes, columns: clusters, entries: number of class elements in cluster.
- Most entries should be on diagonal.

Purity

- of clusters: percentage of elements belonging to dominant class.
- of classes: percentage of elements belonging to dominant cluster.

→ Ignores elements from other classes/clusters

Entropy

- Entropy of cluster *j* $$E_j = \sum_{i=1}^{k} \frac{m_{ij}}{M_j} \log(\frac{m_{ij}}{M_j})$$

mij: number of elements of class *i* in cluster *j*

Mj: number of elements in cluster *j*

Cluster Validation

External Validation Criteria

Rand Index

- Measures the agreement between clusters and classes, considering all pairs of objects.

a: number of pairs of objects in same class and same cluster

b: number of pairs of objects in different class and different cluster

c: number of pairs of objects in same class and different cluster

d: number of pairs of objects in different class and same cluster

$$R = \frac{a + b}{a + b + c + d}$$

Representative-based Clustering

Basics

Goal

A (disjoint) partitioning into *k* clusters with minimal costs.

Local optimization method

- Choose *k* initial cluster representatives.
- Optimize these representatives iteratively.
- Assign each object to its most similar cluster representative.

Types of cluster representatives

- Mean of a cluster (*construction of central points*)
- Medoid of a cluster (*selection of representative points*)

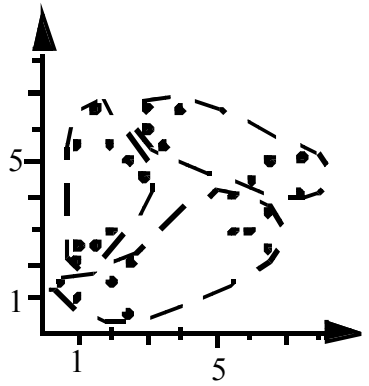
Construction of Central Points

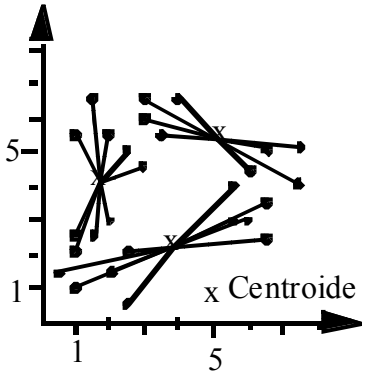
Example

Cluster

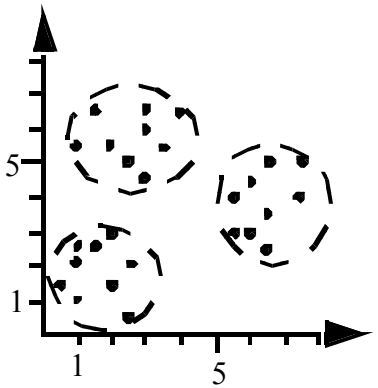
Cluster Representatives

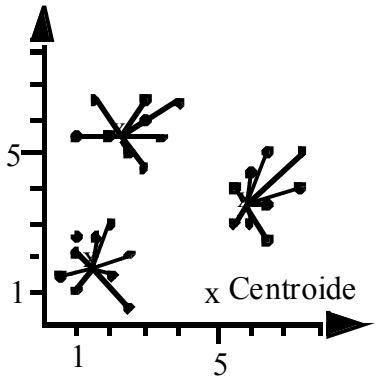
bad clustering





optimal clustering





Construction of Central Points

Basics [Forgy 1965]

- Objects are points $p=(x^p_1, ..., x^p_d)$ in an Euclidean vector space.
- Euclidean distance
- *Centroid* μ_C: mean vector of all objects in cluster C
- *Measure for the cost* (compactness) *of a cluster C*

$$TD^2(C) = \sum_{p \in C} dist(p, \mu_C)^2$$

- *Measure for the cost* (compactness) *of a clustering*

$$TD^2 = \sum_{i=1}^{k} TD^2(C_i)$$

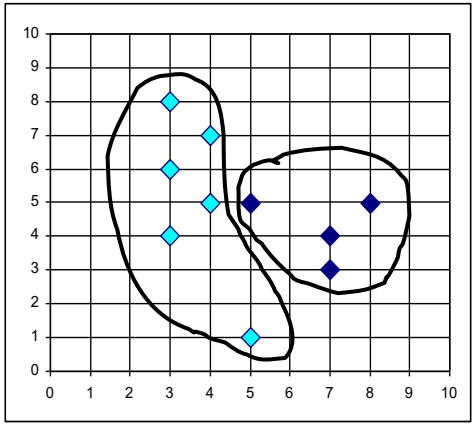
Construction of Central Points

Algorithm

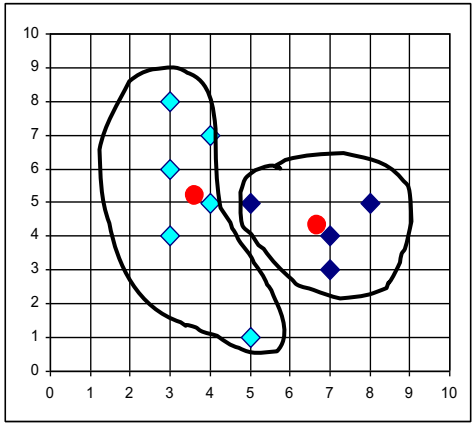
```
ClusteringByVarianceMinimization(dataset D, integer k)
  Initialize the set C'={C_1, ..., C_k} of the centroids
   of the k clusters;
  C = {};
  repeat until C = C'
     C = C';
     form k clusters by assigning each object to the
      closest centroid from C;
     re-calculate the set C'={C'_1, ..., C'_k} of the
      centroids for the newly determined clusters;
  return C;
```

Construction of Central Points

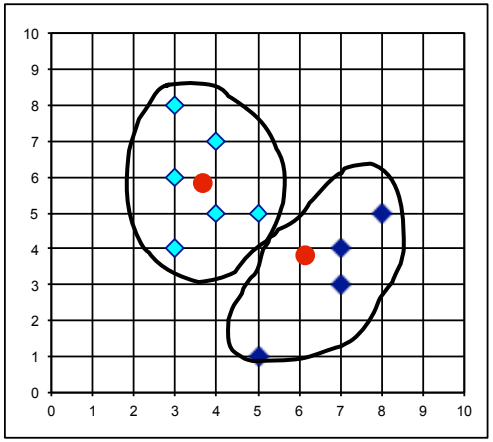
Example



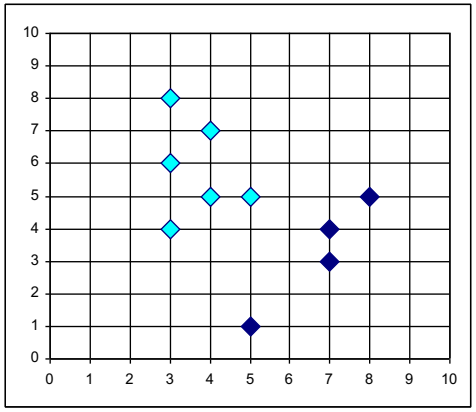
calculate the new centroids



assign to the closest centroid



calculate the new centroids



Construction of Central Points

Variants of the Basic Algorithm

k-means [MacQueen 67]

- Idea: the relevant centroids are updated immediately when an object changes its cluster membership.
- *K*-means inherits most properties from the basic algorithm.
- *K*-means depends on the order of objects.

ISODATA

- Based on *k*-means.
- Post-processing of the resulting clustering by
  - elimination of very small clusters,
  - merging and splitting of clusters.
- User has to provide several additional parameter values.

Construction of Central Points

Discussion

Pros

Cons

Selection of Representative Points

Basics [Kaufman & Rousseeuw 1990]

- Assumes only distance function for pairs of objects, no Euclidean vector space.
- Less sensitive to outliers.
- *Medoid*: a representative element of the cluster (representative point)

  that minimizes the sum of the distances from the other cluster elements
- *Measure for the cost* (compactness) *of a cluster C* $TD(C) = \sum_{p \in C} dist(p, m_C)$
- *Measure for the cost* (compactness) *of a clustering*

$$TD = \sum_{i=1}^{k} TD(C_i)$$

- Search space for the clustering algorithm:
  all subsets of cardinality k of the dataset D with $|D| = n$

Runtime complexity $O(n^k)$

Selection of Representative Points

Overview of the Algorithms

PAM [Kaufman & Rousseeuw 1990]

- Greedy algorithm:
  in each step, one medoid is replaced by one non-medoid.
- Always select the pair (medoid, non-medoid) which implies the largest reduction of the cost TD.

CLARANS [Ng & Han 1994]

Two additional parameters: *maxneighbor* and *numlocal*

- At most *maxneighbor* many randomly chosen pairs (medoid, non-medoid) are considered.
- The first replacement reducing the *TD*-value is performed.
- The search for *k* „optimum“ medoids is repeated *numlocal* times.

Selection of Representative Points

Algorithm PAM

```
PAM(dataset D, integer k, float dist)
  initialize the k medoids and assign all points;
  compute the cost TD of the initial clustering;
  TD_Update := −∞;
  while TD_Update < 0 do
     for each pair (medoid M, non-medoid N),
      calculate the value of TD_{N↔M};
     choose the pair (M, N) with minimum value for
      TD_Update := TD_{N↔M} − TD;
     if TD_Update < 0 then
          replace medoid M by non-medoid N;
          record the set of the k current medoids as the
            currently best clustering;
          TD := cost of the best clustering;
  return best k medoids;
```

Selection of Representative Points

Algorithm CLARANS

```
CLARANS(dataset D, integer k, float dist,
        integer numlocal, integer maxneighbor)
  for r from 1 to numlocal do
     choose randomly k objects as medoids;
     i := 0;
     while i < maxneighbor do
           choose randomly(medoid M, non-medoid N);
           calculate TD_Update := TD_{N↔M} − TD;
           if TD_Update < 0 then
             replace M by N;
             TD := TD_{N↔M}; i := 0;
           else i:= i + 1;
     if TD < TD_best then
           TD_best := TD; record the current medoids;
  return current (best) medoids;
```

Selection of Representative Points

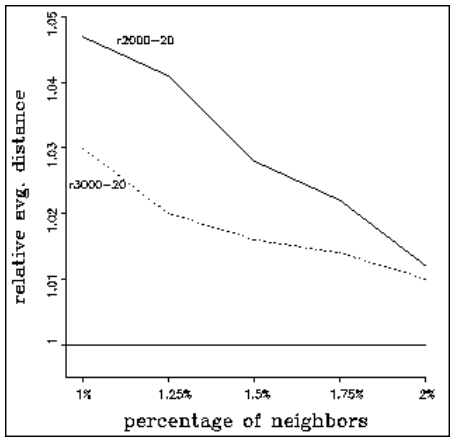
Comparison of PAM and CLARANS

Runtime complexities

- PAM: O($n^3 + k(n-k)^2$ * *#Iterations*)
- CLARANS O(*numlocal* * *maxneighbor* * #replacements * *n*)
  in practice, O(n^2)

Experimental evaluation

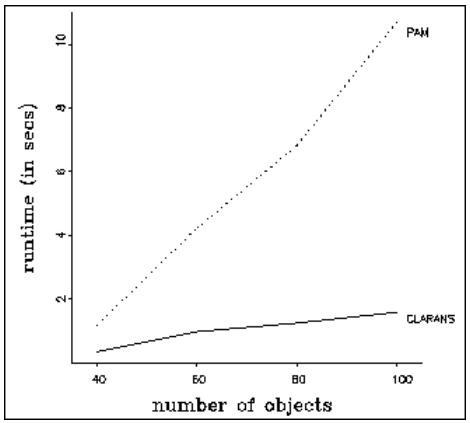
Quality



TD(CLARANS)

TD(PAM)

Runtime



Selection of Representative Points

Discussion

Pros

Cons

Choice of Initial Clusterings

Standard approach

• Draw a random sample of k objects from D and take them as initial cluster representatives.

→ This sample may contain outliers!

Better approach

• Draw a sample S of >> k objects from D, cluster S with k-means, and take the resulting cluster representatives as initial cluster representatives to cluster D.

→ To initialize the clustering of S, draw k objects randomly from S.

- To make the initialization even more robust to outliers, repeat this procedure m times and choose the best set of k cluster representatives.

Choice of Initial Clusterings

Even more sophisticated method [Fayyad, Reina & Bradley 1998]

- Draw independently *m* different samples from *D*.
- Cluster each of these samples.

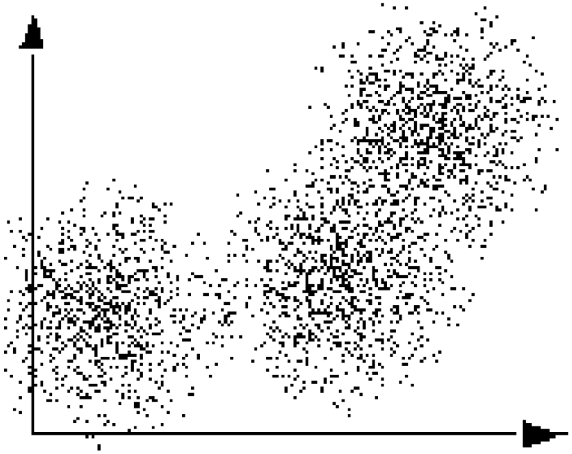
  m different estimates for the *k* cluster means

  $A = (A_1, A_2, \ldots, A_k)$, $B = (B_1, \ldots, B_k)$, $C = (C_1, \ldots, C_k)$, . . .
- Cluster the dataset $DB = A \cup B \cup C \cup \ldots$

  with *m* different initial clusterings *A*, *B*, *C*, . . .
- From the *m* clusterings obtained, choose the representatives of the clustering with the highest quality as initial cluster representatives for the whole dataset *D*.

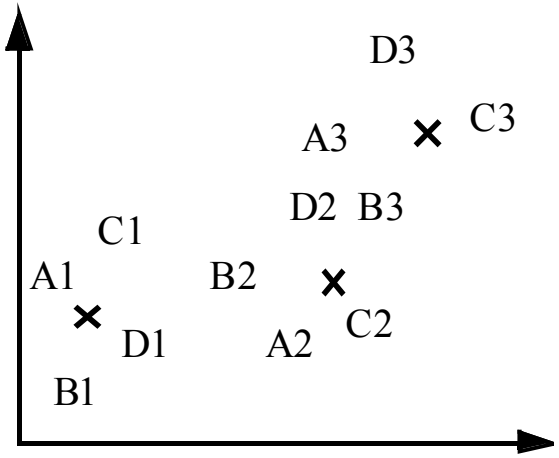
Choice of Initial Clusterings

Example



whole dataset

$k = 3$



DB

from $m = 4$ samples

× *true cluster means*

Choice of Parameter k

Method

- For $k = 2, ..., n-1$, determine one clustering each.
- Choose the clustering with the highest clustering quality.

→ enough to try $k = 2,\ldots,\sqrt{n}$

Measure of clustering quality

- Independent from *k*.
- For *k*-means and *k*-medoid:

  TD^2 and TD decrease monotonically with increasing *k*.

- For EM:

  E increases monotonically with increasing *k*.

Choice of Parameter k

Silhouette-Coefficient [Kaufman & Rousseeuw 1990]

- Measure of clustering quality for *k*-means- and *k*-medoid-methods.
- $a(o)$: distance of object o to its cluster representative
  $b(o)$: distance of object o to the representative of the „second-best“ cluster
- *Silhouette* $s(o)$ of o

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$

  $s(o)$ = -1 / 0 / +1: bad / indifferent / good assignment
- *Silhouette coefficient* s_C of clustering C

  average silhouette over all objects
- Interpretation of silhouette coefficient

  $s_C > 0{,}7$: strong cluster structure,

  $s_C > 0{,}5$: reasonable cluster structure, . . .

Probabilistic Model-based Clustering

EM Clustering [Dempster, Laird & Rubin 1977]

- Objects are points $p=(x^p_1, ..., x^p_d)$ in an Euclidean vector space.
- A cluster is desribed by a probability density distribution.
- Typically: Gaussian distribution (Normal distribution)
- Representation of a cluster C
  - mean μ_C of all cluster points
  - d x d covariance matrix Σ_C for the points of cluster C

$$P(x|C) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_C|}} e^{\frac{1}{2}\cdot(x-\mu_C)^T\cdot(\Sigma_C)^{-1}\cdot(x-\mu_C)}$$

→ EM Clustering algorithm

Probabilistic Model-based Clustering

Basics

- Comparison to k-means

  k-means: (hard) assignment to one cluster based on distance

  EM Clustering: (soft) assignment to all clusters based on membership probability

- Assumption: data has been generated through the following process.

- Generative process

  For i from 1 to n do

  - Sample the cluster j from $P(C_j)$

  - Sample x_i from $P(x_i \mid C_j)$

- Data points are generated independently from the identical distribution (iid).

Probabilistic Model-based Clustering

Basics

- Only data is given. The assignment of points to clusters and the parameters of the generative process are unobserved and need to be estimated from the observed data.
- Parameters of a clustering $M = \{C_1, \ldots, C_k\}$

  $W_c = P(C)$

  μ_C and Σ_C which determine $P(x \mid C)$
- Maximum likelihood estimation

  Determine the unobserved parameters that are most likely to have generated the observed data set.

  Maximize the log of the likelihood of a clustering $M = \{C_1, \ldots, C_k\}$:

$$E(M) = \sum_{x \in D} \log(P(x))$$

Probabilistic Model-based Clustering

Basics

- *Probability density function* of *clustering M*

$$P(x) = \sum_{i=1}^{k} P(C_i)P(x \mid C_i)$$

where $P(x|C) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_C|}} e^{\frac{1}{2}\cdot(x-\mu_C)^T\cdot(\Sigma_C)^{-1}\cdot(x-\mu_C)}$

- Estimation of $P(C_i)$ *as* $W_i = \sum_{x \in D} P(C_i \mid x) \Big/ |D|$

  as the weighted percentage of points that are currently assigned to cluster C_i

- Assignment of points to clusters $P(C_i \mid x) = P(C_i) \cdot \frac{P(x \mid C_i)}{P(x)}$

every point assigned to all clusters with different probabilities

Expectation Maximization

Algorithm

ClusteringByExpectationMaximization (dataset D, integer k)

create an „initial" clustering M = (C_1', ..., C_k');

calculate w_C, μ_C and Σ_C for each cluster C;

repeat // E-step: re-compute cluster assignments

M' := M;

calculate P(x|C_i), P(C_i), P(x), and P(C_i|x) for each object x of D and each cluster C_i;

compute a new clustering M by re-assignment of all objects x according to P(C_i|x);

// M-step: determine new cluster parameters

re-calculate the parameters w_C, μ_C and Σ_C for each cluster C;

until |E(M) - E(M')| < ε;

return M;

Probabilistic Model-based Clustering

Discussion

Pros

Cons

Hierarchical Clustering

Basics

Goal

Construction of a hierarchy of clusters (*dendrogram*).

Dendrogram

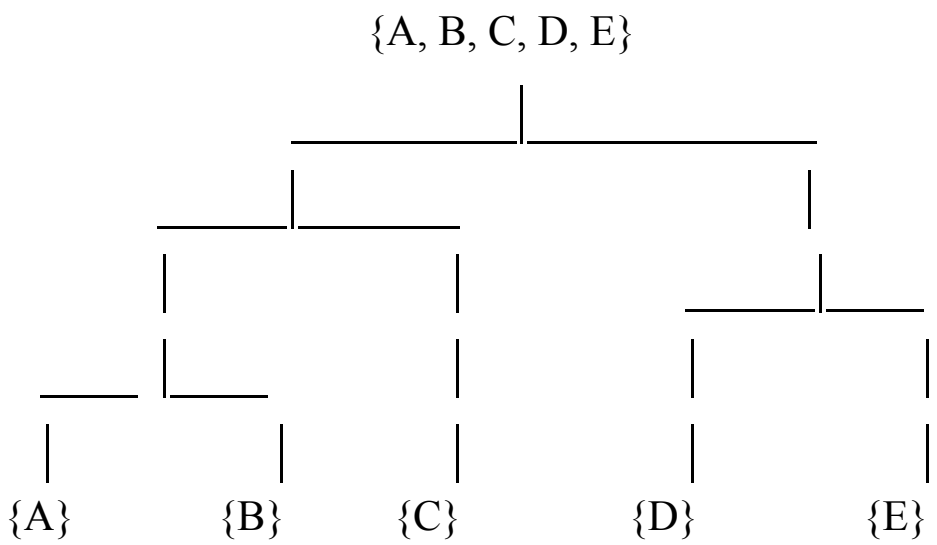
A binary tree of nodes representing clusters, with the following properties:

- Root represents the whole dataset.
- Leaf node represents singleton clusters containing a single object.
- Inner node represents the union of all objects contained in its corresponding subtree.

Hierarchical Clustering

Basics

Example dendrogram



Agglomerative	Divisive
step 4	step 1
step 3	step 2
step 2	step 3
step 1	step 4

Types of hierarchical methods

- Bottom-up construction of dendrogram (*agglomerative*).
- Top-down construction of dendrogram (*divisive*).

Hierarchical Clustering

Algorithmic Scheme [Jain & Dubes 1988]

Agglomerative Hierarchichal Clustering

1. Form initial clusters consisting of a singleton object, and compute the distance between each pair of clusters.
2. Merge the two clusters having minimum distance.
3. Calculate the distance between the new cluster and all other clusters.
4. If there is only one cluster containing all objects:
   Stop, otherwise go to step 2.

A similarity function can be used instead of a distance function. In that case, merge the two clusters that have the maximum similarity.

Hierarchical Clustering

Distance Functions for Clusters

- Let *dist*(*x*,*y*) be a distance function for pairs of objects *x*, *y*.
- Let X, Y be clusters, i.e. sets of objects.

Single-Link

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

Complete-Link

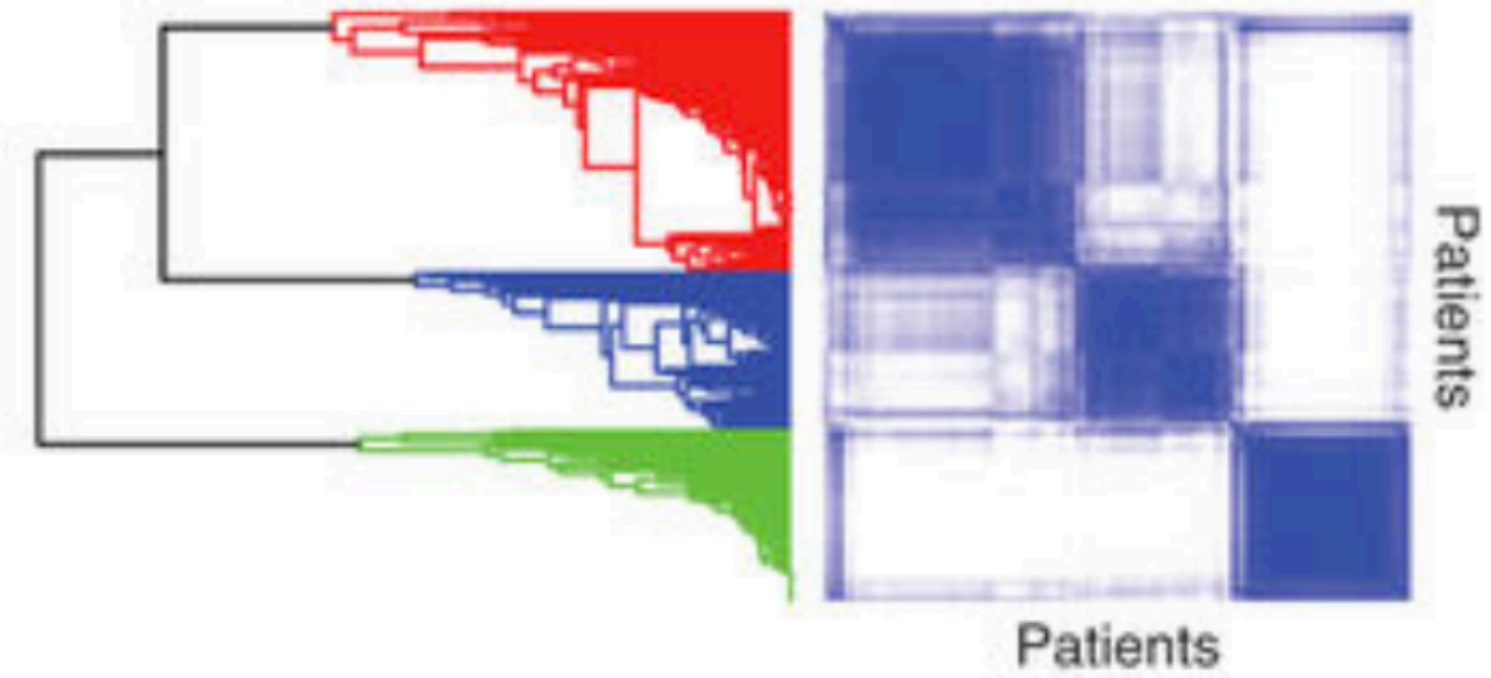
$$dist\_cl(X,Y) = \max_{x \in X, y \in Y} dist(x,y)$$

Average-Link

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

Hierarchical Clustering

Example



- Goal: Identify subtypes of a disease.
- Approach: Cluster patients based on their gene expression data. Similarity of patients can be defined using some Correlation Coefficient.

Hierarchical Clustering

Discussion

Pros

Cons

One Minute Survey

Comparing partitioning clustering algoerithms and hierarchical clustering algorithms,

- What are the advantages of partitioning clustering?
- What are the advantages of hierarchical clustering?

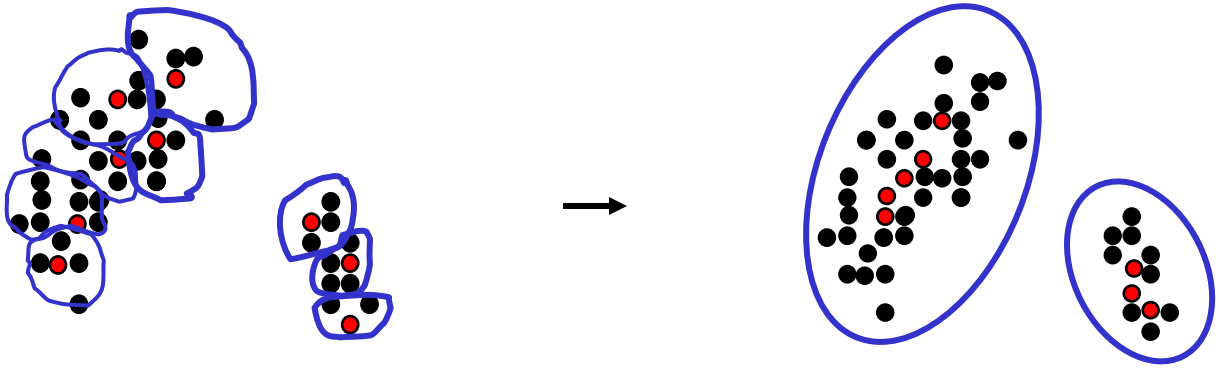
→ Discuss with your neighbor.

→ Share your ideas.

Single-Link and Variants

CURE [Guha, Rastogi & Shim 1998]

- Representation of a cluster
  - partitioning methods: one object
  - hierarchical methods: all objects.
- CURE: representation of a cluster by *c* representatives.
- Representatives are stretched by factor of α w.r.t. the centroid.



Detects non-convex clusters.

Avoids Single-Link effect.

Density-Based Clustering

Introduction

Idea

- Clusters as dense areas in a *d*-dimensional dataspace, separated by areas of lower density.

Requirements for density-based clusters

- For each cluster object, the local density exceeds some threshold.
- The set of objects of one cluster must be spatially connected.

Strenghts of density-based clustering

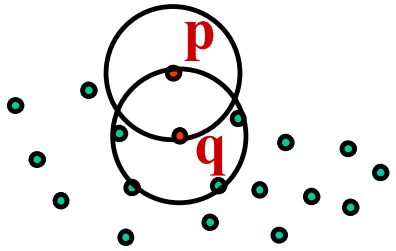
- Clusters of arbitrary shape
- Robust to noise
- Efficiency

Density-Based Clustering

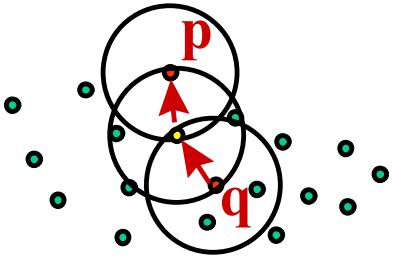
Concepts [Ester, Kriegel, Sander & Xu 1996]

- Object o $\in D$ is *core object* (w.r.t. D):

  $|N_\varepsilon(o)| \geq MinPts$, with $N_\varepsilon(o) = \{o' \in D \mid dist(o, o') \leq \varepsilon\}$.



- Object $p \in D$ is *directly density-reachable* from $q \in D$ w.r.t. ε and *MinPts:* $p \in N_\varepsilon(q)$ and q is a core object (w.r.t. D).
- Object p is *density-reachable* from q: there is a chain of directly density-reachable objects between q and p.

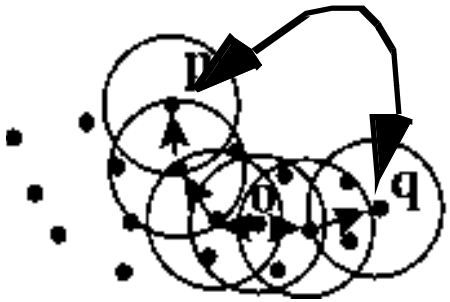


border object: no core object,
but density-reachable from some core object

Density-Based Clustering

Concepts

- Density reachability is not a symmetric relationship. But symmetry is desirable to make the clustering independent from the order of visiting objects.
- Objects p and q are *density-connected*: both are density-reachable from a third object o.



- Density connectedness is a a symmetric relationship.
- *Cluster C* w.r.t. ε and *MinPts*: a non-empty subset of D satisfying

  Maximality: $\forall p,q \in D$: if $p \in C$, and q density-reachable from p, then $q \in C$.

  Connectivity: $\forall p,q \in C$: p is density-connected to q.

Density-Based Clustering

Concepts

- Clustering

  A *density-based clustering CL* of a dataset *D* w.r.t. ε and *MinPts* is the set of all density-based clusters w.r.t. ε and *MinPts* in *D*.

- The set $Noise_{CL}$ („*noise*“) is defined as the set of all objects in *D* which do not belong to any of the clusters.

- Property

  Let *C* be a density-based cluster and $p \in C$ a core object. Then: $C = \{o \in D \mid o$ density-reachable from p w.r.t. ε and *MinPts*$\}$.

Density-Based Clustering

Algorithm DBSCAN

```
DBSCAN(dataset D, float ε, integer MinPts)
  all objects of D are initialized as not assigned to any
    cluster (i.e., noise);
  ClusterId := 1;
  for each object o in D do
     if o is not yet assigned to any cluster then
        if o is a core object then
           Determine all objects that are density-reachable
             from o w.r.t. ε and MinPts and assign them to
             cluster ClusterId;
           ClusterId:= ClusterId + 1;
  return D with cluster assignments for each object
```

Density-Based Clustering

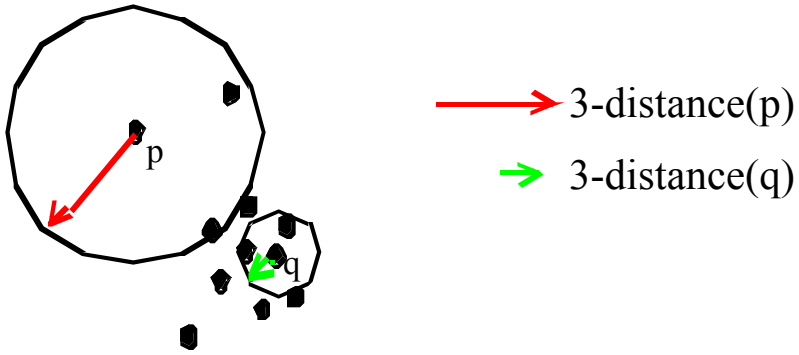
Algorithm DBSCAN (Graph-based Formulation)

```
DBSCAN(dataset D, float ε, integer MinPts)
  Determine core objects, border objects and noise objects
    of D for parameters ε and MinPts;
  Create graph in which nodes represent objects and pairs
    of core objects are connected if they are within ε of
     one another;
  Determine connected components in graph;
  Assign each border object to the connected component
    with which it is best connected;
 return objects of each connected component as a cluster;
```

Density-Based Clustering

Choice of Parameters [Schubert et al 2017]

- Cluster: density above the „minimum density“ defined by ε and *MinPts*.
- Wanted: the density of the cluster with the lowest density. All objects with higher density should belong to one of the clusters.
- Heuristic method: consider the distances to the *k*-nearest neighbors.

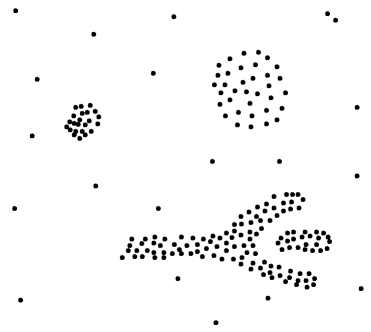


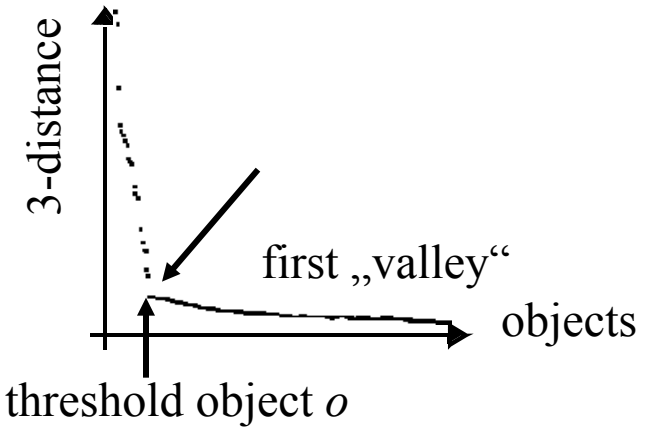
- Function *k-distance*: distance of an object to its *k*-nearest neighbor.
- *k-distance-diagram*: *k*-distances in descending order.

Density-Based Clustering

Choice of Parameters

Example





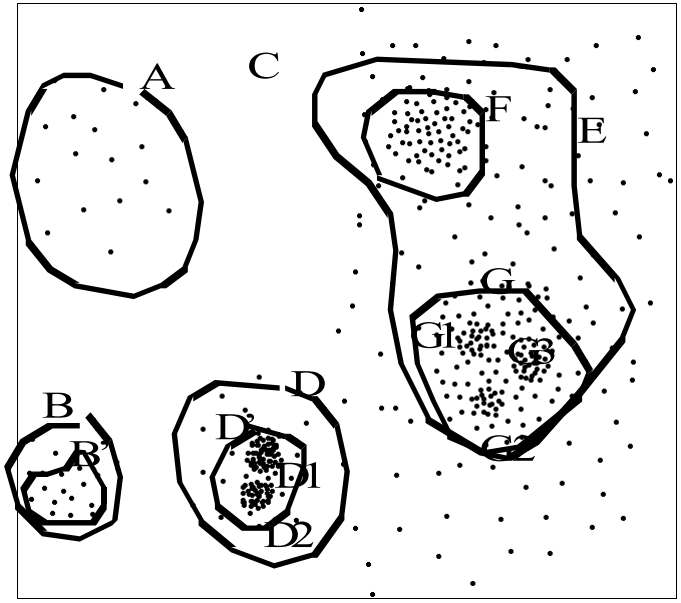
Heuristic Method

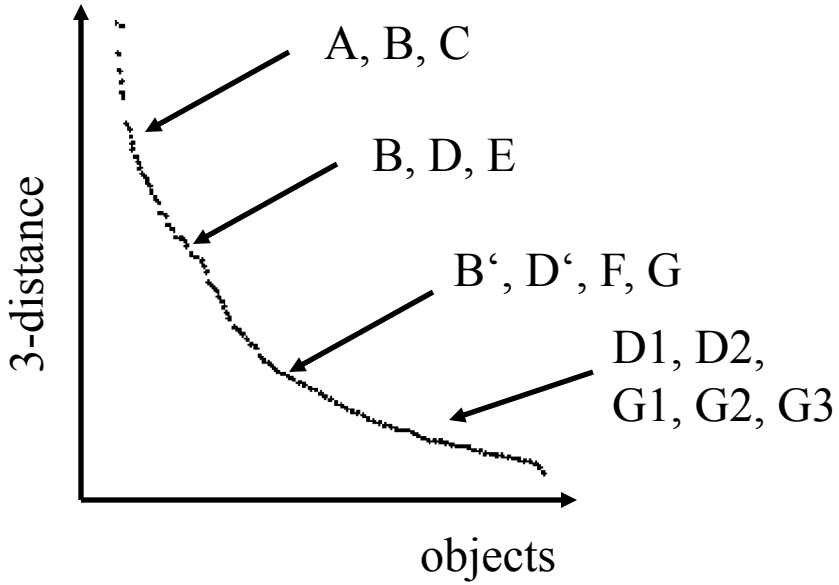
- User specifies a value for *k* (Default is $k = 2*d - 1$), *MinPts* := *k*+1.
- System calculates the *k*-distance-diagram for the dataset and visualizes it.
- User chooses a threshold object *o* from the *k*-distance-diagram, ε := *k*-distance(*o*).

Density-Based Clustering

Problems with Choosing the Parameters

- Hierarchical clusters
- Significantly differing densities in different areas of the dataspace.
- Clusters and noise are not well-separated.

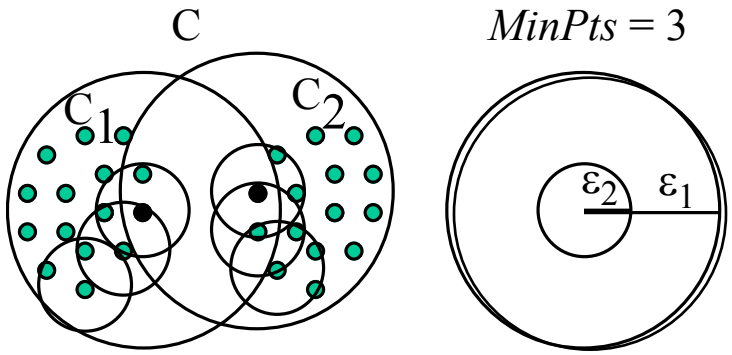




Hierarchical Density-Based Clustering

OPTICS [Ankerst, Breunig, Kriegel & Sander 1999]

- For constant *MinPts*-value, density-based clusters w.r.t. a *smaller* ε are completely contained within density-based clusters w.r.t. a *larger* ε.



- The clusterings for different density parameters can be determined simultaneously in a single scan: first dense sub-cluster, then less dense rest-cluster.

→ The relevant density parameters are typically unknown.

- OPTICS does not generate a dendrogramm, but a graphical visualization of the hierarchical cluster structure considering all relevant density parameters.

Hierarchical Density-Based Clustering

Concepts

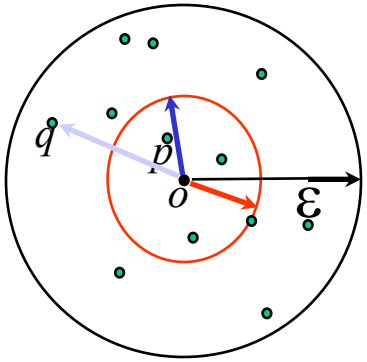
Core distance of object p w.r.t. ε and *MinPts*

$$Core\,Distance_{\varepsilon,MinPts}(o) = \begin{cases} UNDEFINED,\ if\ |N_\varepsilon(o)| < MinPts \\ MinPtsDistance(o),\ else \end{cases}$$

Reachability distance of object p relative to object o

$$Reachability\ Distance_{\varepsilon,MinPts}(p,o) = \begin{cases} UNDEFINED,\ if\ |N_\varepsilon(o)| < MinPts \\ \max\{Core\ Distance(o), dist(o,p)\},\ else \end{cases}$$

$MinPts = 5$



Core distance(o)

Reachability distance(p,o)

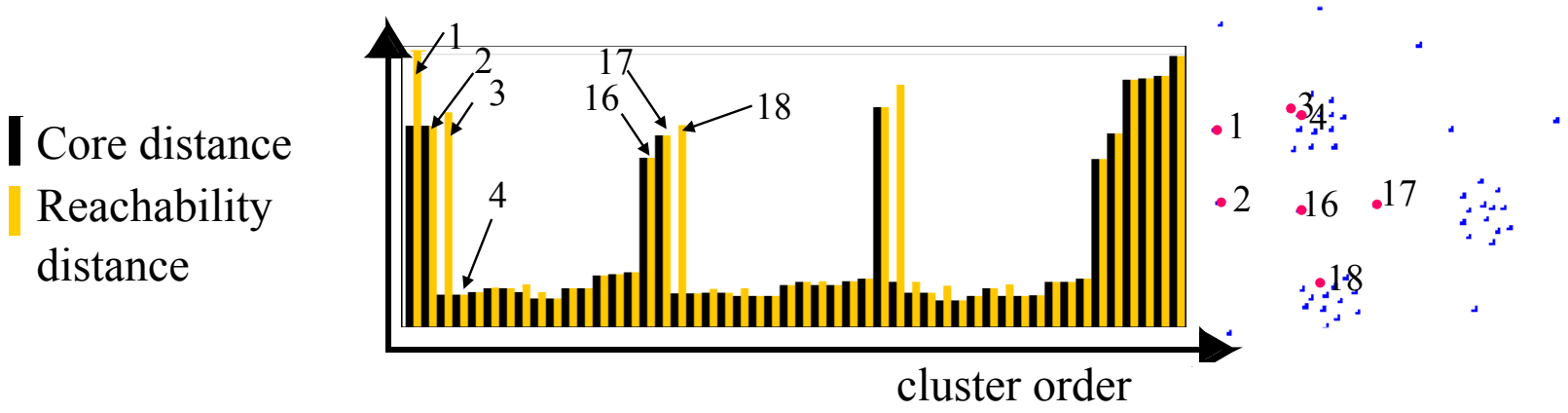
Reachability distance(q,o)

Hierarchical Density-Based Clustering

Cluster Order

- OPTICS does not directly return a (hierarchichal) clustering, but orders the objects according to a „cluster order“ w.r.t. ε and *MinPts*.
- *Cluster order* w.r.t. ε and *MinPts*
  - Start with an arbitrary object.
  - Until all objects have been visited, visit the object that has the minimum reachability distance from the set of already visited objects.

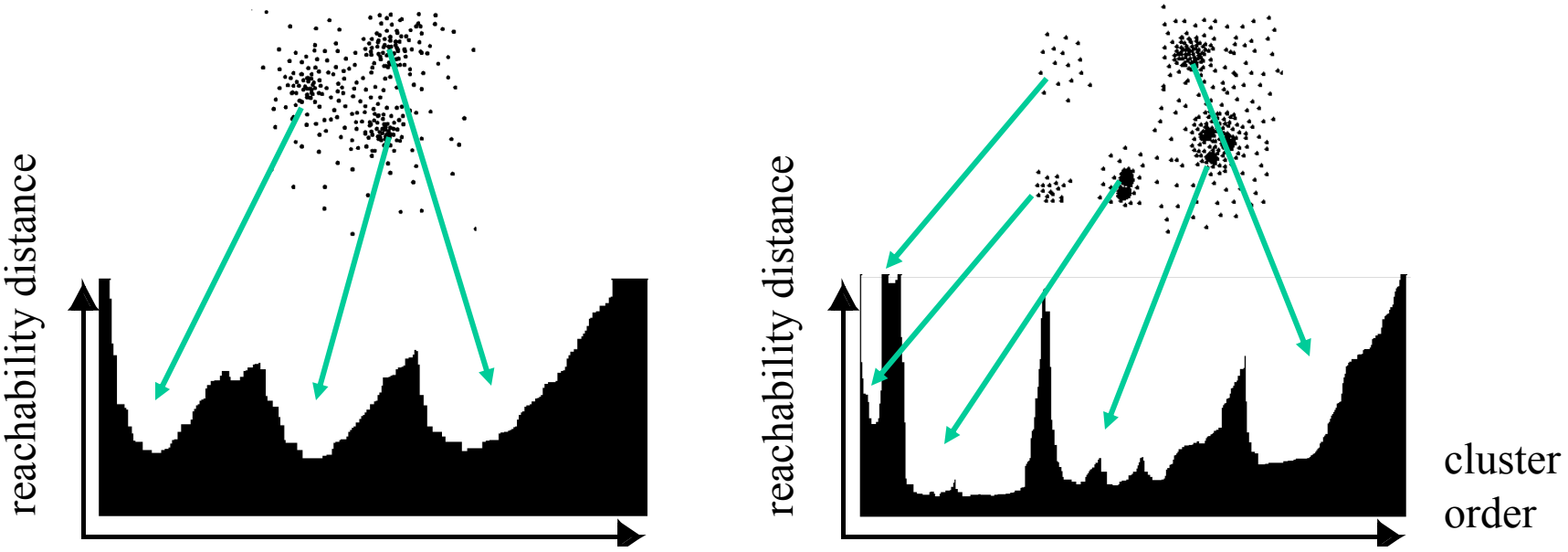
$$\textit{Reachability Distance}_{\varepsilon,MinPts}(p,\{p_1,\ldots,p_k\}) = \min\{\textit{Reachability Distance}_{\varepsilon,MinPts}(p,p_i) \mid 1 \le i \le k\}$$



Hierarchical Density-Based Clustering

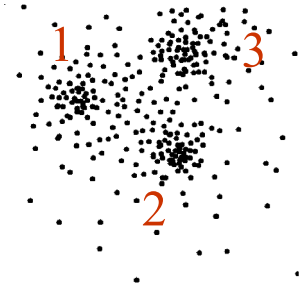
Reachability Diagram

- Depicts the reachability distances (w.r.t. ε and *MinPts*) of all objects in a bar diagram.
- With the objects ordered according to the cluster order.

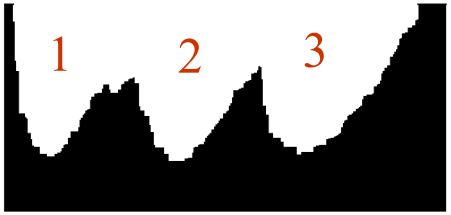


Hierarchical Density-Based Clustering

Sensitivity of Parameters



MinPts = 10, ε = 10

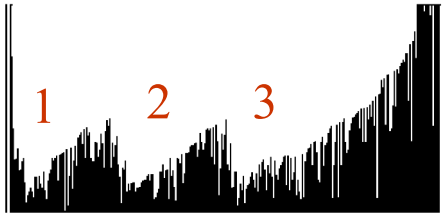


"optimum" parameters

MinPts = 10, ε = 5

smaller ε

MinPts = 2, ε = 10



smaller *MinPts*

Cluster order is robust against changes of the parameters.

Good results as long as parameters „large enough".

Hierarchical Density-Based Clustering

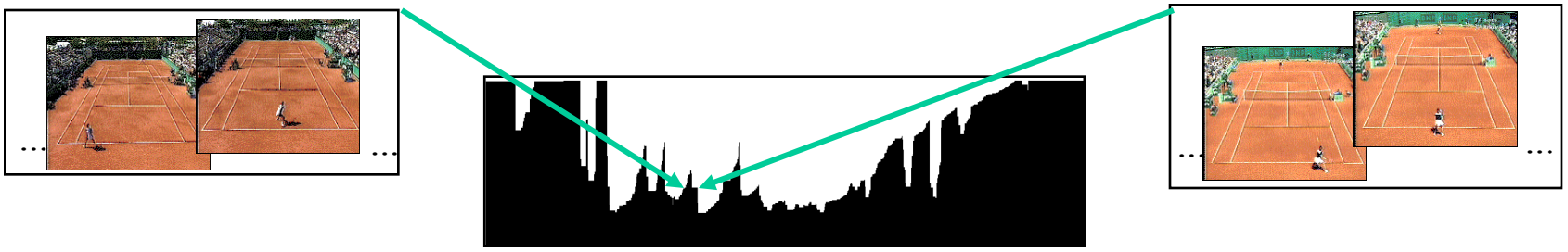
Heuristics for Setting the Parameters

ε

- Choose largest *MinPts*-distance in a sample or
- calculate average *MinPts*-distance for uniformly distributed data.

MinPts

- Smooth reachability-diagram.
- Avoid "single-link" effect.



Hierarchical Density-Based Clustering

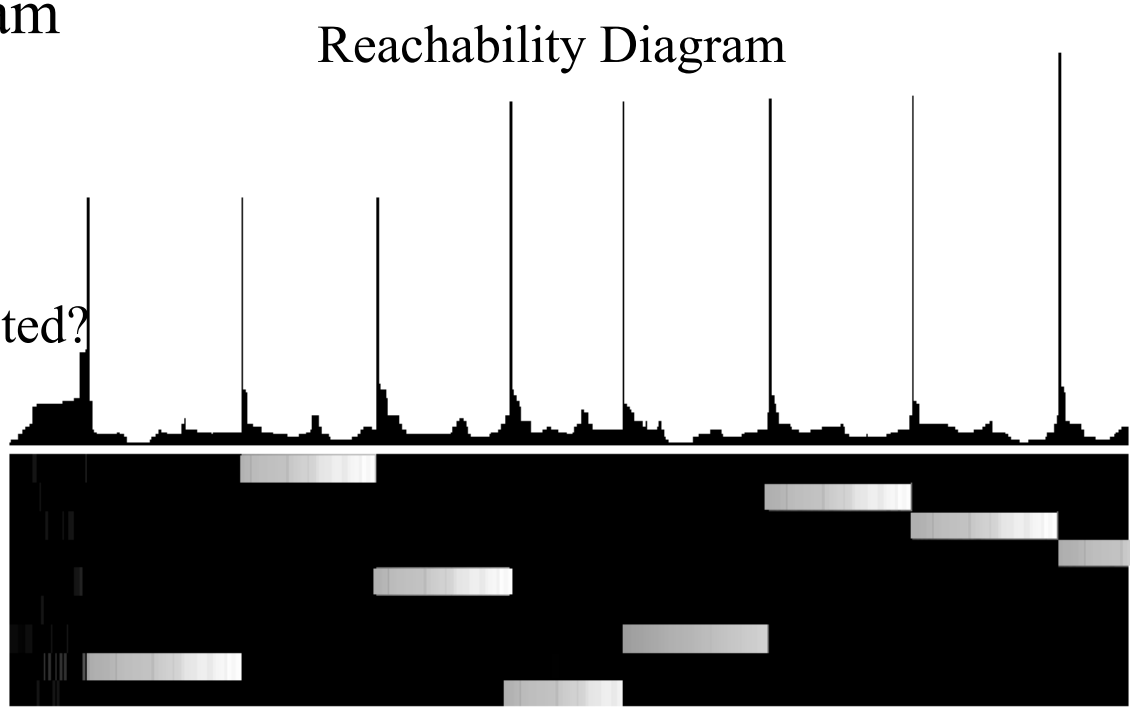
Manual Cluster Analysis

Based on Reachability Diagram

- Are there clusters?
- How many clusters?
- How large are the clusters?
- Are the clusters hierarchically nested?

Based on Attribute Diagram

- Why do clusters exist?
- What attributes allow to distinguish the different clusters?

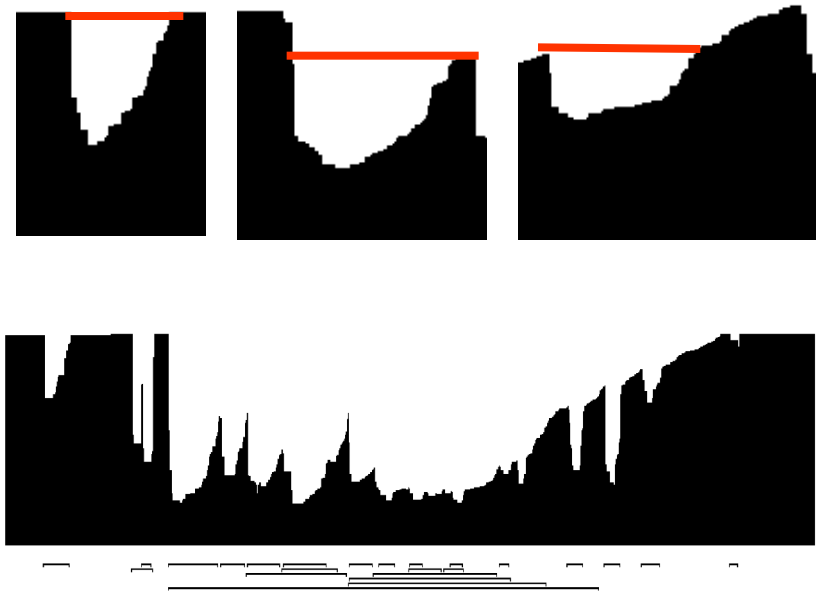


Hierarchical Density-Based Clustering

Automatic Cluster Analysis

ξ-Cluster

- Subsequence of the cluster order.
- Starts in an area of ξ-steep *decreasing* reachability distances.
- Ends in an area of ξ-steep *increasing* reachability distances at approximately the same absolute value.
- Contains at least *MinPts* objects.



Algorithm

- Determines all ξ-clusters.
- Marks the ξ-clusters in the reachability diagram.
- Runtime complexity O(*n*).

Density-Based Clustering

Discussion

Pros

Cons

Non-Negative Matrix Factorization

Introduction [Lee & Seung 2001]

- Nonnegative matrix factorization (*NMF*) is a dimensionality reduction method that is tailored to clustering.
- Suitable for matrices that are non-negative and sparse.
  E.g.: document-term matrix of term frequencies
  → Most term frequencies are 0.
- Embeds the data into a latent, lower-dimensional space that makes it more amenable to clustering.
- The basis system of vectors and the coordinates of the data objects in this system are nonnegative, which makes the solution interpretable.

Non-Negative Matrix Factorization

Introduction

$$D \approx UV^T$$

D: n x d data matrix of n objects with d attributes

V: d x k matrix of the k basis vectors in terms of the original lexicon

U: n x k matrix of k-dimensional coordinates of the rows of D in the transformed basis system

→ $k << d$

→ Basis vectors represent topics / clusters of attributes.

- Objective

$$\underset{U,V}{\operatorname{argmin}} \left\| D - UV^T \right\|^2 \; subject\ to\ U \geq 0, V \geq 0$$

where $\|\cdot\|^2$ denotes the squared Frobenius norm.

Non-Negative Matrix Factorization

Alternating Non-negative Least Squares

Initialize $U^1 \geq 0,\ V^1 \geq 0$

Repeat until convergence

$$U^{k+1} = \underset{U \geq 0}{\operatorname{argmin}}\, f(U, V^k)$$

$$V^{k+1} = \underset{V \geq 0}{\operatorname{argmin}}\, f(U^{k+1}, V)$$

Where $f(U,V) = \left\| D - UV^T \right\|^2$

- The two non-negative least square subproblems are convex.
- Can be solved using projected Newton's method or projected gradient methods for bound-constrained optimization.

Non-Negative Matrix Factorization

More Details

- To avoid overfitting, add a regularizer to the objective function, e.g. $\lambda(\|U\|^2 + \|V\|^2)$.
- Compared to SVD, NMF is harder to optimize because of the non-negativity constraint.
- But SVD may produce negative entries for U,V which makes it less useful for clustering.
- PLSA can be considered as a variant of NMF that interprets the nonnegative elements of the (scaled) matrix as probabilities and maximizes the likelihood of D under a probabilistic generative model.

Non-Negative Matrix Factorization

Discussion

Pros

Cons

Cluster Ensembles

Motivation

- Different clustering algorithms produce many different clusterings.
- Cluster validation is typically hard.
- None of the many alternative clusterings is the "true" clustering, but they all capture some aspects of the cluster structure.
- The goal of cluster ensembles is to combine multiple clusterings to create a more robust clustering.
- Do not consider the attributes/features of the objects, but only the structure of the clusterings.

→ Also called consensus clustering or multi-view clustering.

Cluster Ensembles

Selecting Ensemble Components

- Model-based ensembles
  - clusterings obtained from different clustering algorithms, e.g. partitioning, hierachical and density-based algorithms
  - clusterings obtained from same clustering algorithm with different parameter settings
    e.g. different numbers of clusters, different density threshold
- Data-based ensembles
  - select different subsets of the data
  - select different subsets of the set of dimensions.

Cluster Ensembles

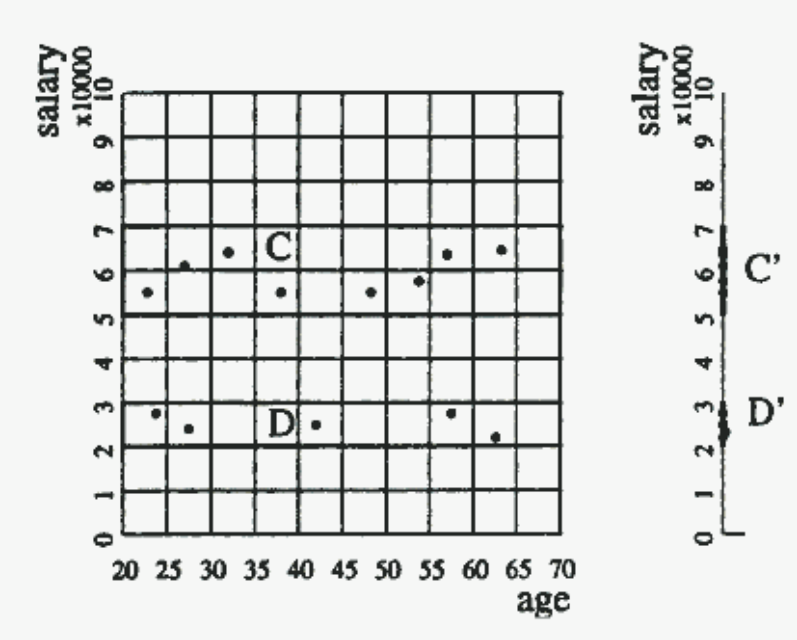
Combining Ensemble Components

- Graph partitioning
  vertex = object, edge = connects objects appearing together in a cluster, edge weight = number of shared clusters
  apply graph partitioning algorithm such as Min-cut
- Hypergraph partitioning
  vertex = object, hyperedge = cluster
  apply hypergraph partitioning algorithm such as HMETIS
- Distance-based clustering
  similarity of two objects: percentage of shared clusters
  apply distance-based algorithm such as hierarchical agglomerative clustering.

Clustering High-Dimensional Data

Curse of Dimensionality

- In high-dimensional data, the (average) pairwise distances are big and rather uniformly distributed.
- Clusters only in lower-dimensional subspaces.



clusters only in 1-dimensional subspace „salary“

Subspace Clustering

CLIQUE [Agrawal et al1998]

- *Cluster*: „dense area“ in dataspace.
- Density-threshold τ

  region is *dense*, if it contains more than τ objects.
- Grid-based approach

  - Each dimension is divided into ξ intervals.
  - Cluster is union of connected dense regions (region = grid cell).
- Phases

  1. Identification of subspaces with clusters.
  2. Identification of clusters.
  3. Generation of cluster descriptions

Subspace Clustering

Identification of Subspaces with Clusters

- Task: detect *dense base* regions.
- Naive approach:

  calculate histograms for all subsets of the set of dimensions.

  ⇒ Infeasible for high-dimensional datasets (O (2^d) for d dimensions).
- Greedy algorithm (Bottom-Up)

  Start with the empty set,

  add one more dimension at a time.
- *Monotonicity property*

  If a region R in k-dimensional space is dense, then each projection of R in (k-1)-dimensional subspace is dense as well (more than τ objects).

  ⇒ If monotonicity property violated, prune candidate region.

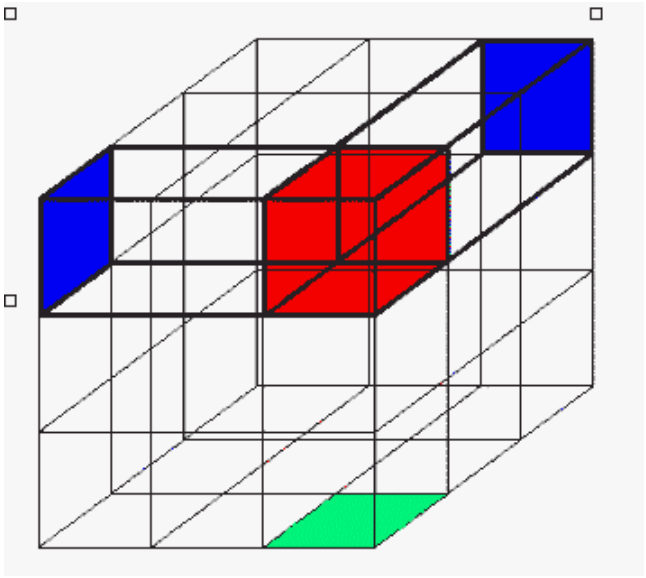
Subspace Clustering

Identification of Subspaces

```
CLIQUE(dataset D, integer ξ, integer τ)
Partition all dimensions into ξ intervals;
Determine set D1 of 1-dimensional τ-dense regions;
K:=2;
while Dk-1 <> {} do
  k-dimensional candidate regions Ck := join pairs of
    (r1,r2) from Dk-1;
  Prune elements from Ck with a k-1 dimensional
    projection not in Dk-1;
  for each object o in D do
    for each region r in Ck do
      if o in r then increment counter of r;
  Dk:= all elements in Ck with counter ≥ τ;
  k:=k+1
```

Subspace Clustering

Example



2-dim. dense regions

3-dim. candidate region

2-dim. region to be tested

- Runtime complexity of greedy algorithm $O(c^k + n \cdot k)$
  for n database objects and k = maximum dimension of a dense region.
- Heuristic reduction of the number of candidate regions
  application of the „Minimum Description Length“- principle.

Subspace Clustering

Identification of Clusters

- Task: find *maximal* sets of *connected* dense base regions.
- Given: all dense base regions in a *k*-dimensional subspace.
- „Depth-first“-search of the following graph (search space)

  nodes: dense base regions,

  edges: joint edges / dimensions of the two base regions.
- Runtime complexity

  Dense base regions in main memory (e.g. hash tree).

  For each dense base region, test 2 *k* neighbors.

  $\Rightarrow$ number of accesses of data structure: 2 *k n*

Subspace Clustering

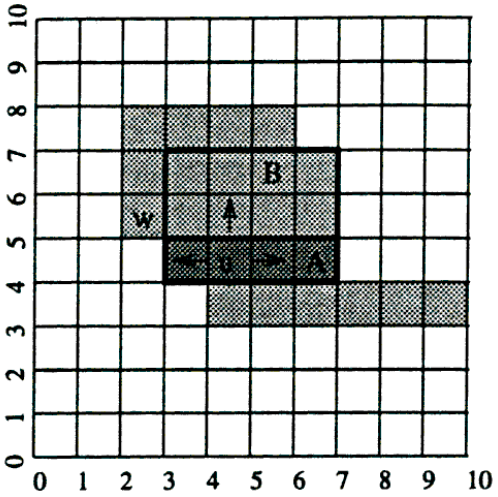
Generation of Cluster Descriptions

- Given: a cluster, i.e. a set of connected dense base regions.
- Task: find optimal cover of this cluster

  by a set of hyperrectangles.
- Standard methods

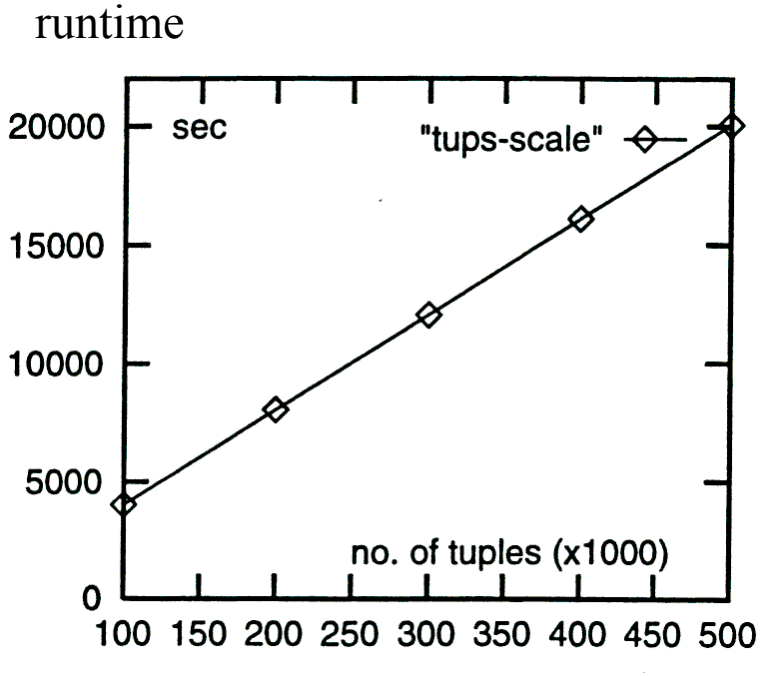
  Infeasible for large values of d .

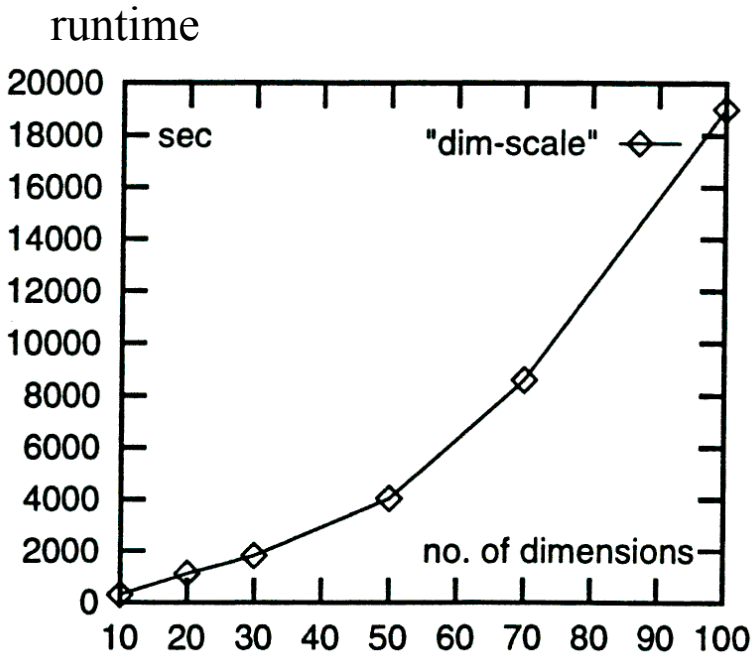
  The problem is NP-complete.
- Heuristic method
  1. Cover the cluster by maximal regions.
  2. Remove redundant regions.



Subspace Clustering

Experimental Evaluation





Runtime complexity of CLIQUE

Linear in *n* , superlinear in *d,* exponential in dimensionality of clusters

Subspace Clustering

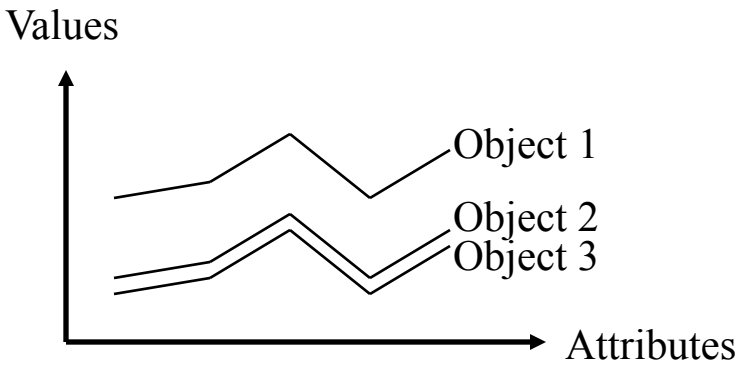
Discussion

Pros

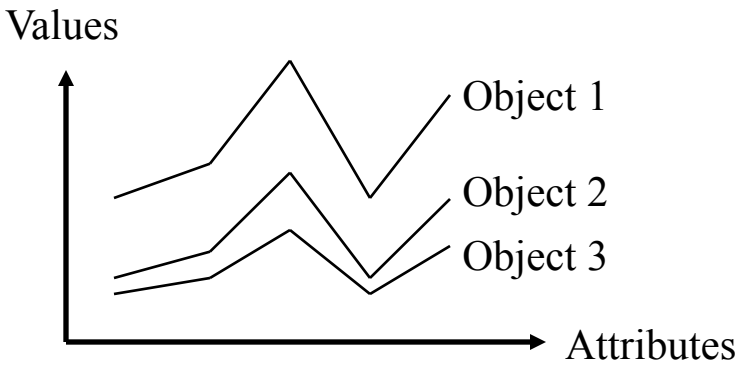
Cons

Subspace Clustering

Pattern-Based Subspace Clusters



Shifting pattern
(in some subspace)



Scaling pattern
(in some subspace)

Such patterns cannot be found using existing subspace clustering methods since

- these methods are distance-based,
- the above points are not close enough.

Subspace Clustering

δ-pClusters [Wang, Wang, Yang &Yu 2002]

- *O: subset of DB objects, T: subset of attributes*
- $x, y \in O$, $a, b \in T$, d_{xa} : value of object x on attribute a

$$pScore\left(\begin{bmatrix} d_{xa} & d_{xb} \\ d_{ya} & d_{yb} \end{bmatrix}\right) = |(d_{xa} - d_{xb}) - (d_{ya} - d_{yb})|$$

$$= |(d_{xa} - d_{ya}) - (d_{xb} - d_{yb})|$$

- (O,T) is a *δ-pCluster*, if for any 2 x 2 submatrix X of (O,T)

$$pScore(X) \leq \delta \text{ for a given } \delta \geq 0$$

- Monotonicity property

If (O,T) is a δ-pCluster and $O' \subseteq O$, $T' \subseteq T$,
then (O',T') is also a δ – pCluster

Subspace Clustering

Problem

- Given δ, *nc* (minimal number of columns), *nr* (minimal number of rows), find all pairs (*O*,*T*) such that
  - (O,T) is a δ–pCluster
  - $|O| \geq nr$
  - $|T| \geq nc$
- For δ–pCluster (*O*,*T*), *T* is a *maximum dimension set* (*MDS*) if there does not exist

$$T' \supset T \text{ such that } (O,T') \text{ is also a } \delta\text{-pCluster}$$

- $S(x,y,T) = \{d_{xa} - d_{ya} \mid a \in T\}$
- Objects *x* and *y* form a δ–pCluster on *T* iff the difference between the largest and smallest value in S(x,y,T) is below δ.

Subspace Clustering

Algorithm

- $\vec{S}(x, y, T) = s_1, \ldots, s_k$ where $s_i \in S(x, y, T)$ and $s_i \leq s_j$ for $i < j$
- *Given A,* $T \subseteq A$ is an *MDS* of x and y iff
  $\vec{S}(x, y, T) = s_i \ldots s_j$ is a contiguous subsequence of $\vec{S}(x, y, A)$ and $s_j - s_i \leq \delta$, $s_{j+1} - s_i > \delta$ and $s_j - s_{i-1} > \delta$.
- Pairwise clustering of x and y:
  - Compute $\vec{S}(x, y, T)$.
  - Identify all subsequences with the above property.

  Ex.: -3 -2 -1 6 6 7 8 8 10, $\delta = 2$

Subspace Clustering

Algorithm

- For every pair of objects (and every pair of colums), determine all MDSs.
- Prune those MDSs.
- Insert remaining MDSs into prefix tree. All nodes of this tree represent candidate clusters (O,T).
- Perform post-order traversal of the prefix tree. For each node, detect the δ–pCluster contained. Repeat until no nodes of depth $\geq n_c$ are left.
- Runtime comlexity $O(M^2 N \log N + N^2 M \log M)$ where M denotes the number of columns and N denotes the number of rows.

Projected Clustering

PROCLUS [Aggarwal et al 1999]

- *Cluster*: $C_i = (P_i, D_i)$

  $P_i \subseteq DB$ and $D_i \subseteq D$ (set of all dimensions)

  Cluster represented by a medoid.

- *Clustering*: $\{C_1, \cdots, C_k, O\}$

  k: user-specified number of clusters

  l: user-specified average number of dimensions per cluster

  O: outliers that are too far away from any of the clusters

- Phases

  1. Initialization
  2. Iteration
  3. Refinement

Projected Clustering

Initialization Phase

- Set of *k* medoids is *piercing*:

  each of the medoids is from a different (true) cluster.

- Objective

  Find a small enough superset of a piercing set

  that allows an effective second phase.

- Method

  Choose random sample *S* of size $A \cdot k$.

  Iteratively choose $B \cdot k$ points from *S* where $B >> 1.0$

  that are far away from already chosen points (yields set *M*).

Projected Clustering

Iteration Phase

- Approach: Local Optimization (Hill Climbing).
- Choose k medoids randomly from M as M_{best}.
- Perform the following iteration step

  Determine the „bad“ medoids in M_{best}.

  Replace them by random elements from M, obtaining $M_{current}$.

  Determine the $k \cdot l$ best dimensions for the k medoids in $M_{current}$.

  Form k clusters, assigning all points to the closest medoid.

  If clustering $M_{current}$ is better than clustering $M_{best,}$ then set M_{best} to $M_{current}$.
- Terminate when M_{best} does not change after a certain number of iterations.

Projected Clustering

Iteration Phase

Determine the $k \cdot l$ best dimensions for the k medoids in $M_{current}$

- Determine the *locality* L_i of each medoid m_i:

  points within $\delta_i = \min_{j \neq i} dist(m_i, m_j)$ from m_i .

- Measure the average distance $X_{i,j}$ from m_i along dimension j in L_i .
- For m_i , determine the set of dimensions j for which $X_{i,j}$ is as small as possible compared to statistical expectation ($Y_i = (\sum_{j=1}^{d} X_{i,j}) \Big/ d$).
- Two constraints:

  Total number of chosen dimensions equal to $k \cdot l$.

  For each medoid, choose at least 2 dimensions.

Projected Clustering

Iteration Phase

Forming clusters in $M_{current}$

- Given the $k \cdot l$ dimensions chosen for $M_{current}$.
- Let D_i denote the set of dimensions chosen for m_i.
- For each point p and for each medoid m_i,
  compute the distance from p to m_i
  using only the dimensions from D_i.
- Assign p to the closest m_i.

Normalize the distances for different cardinalities of D_i

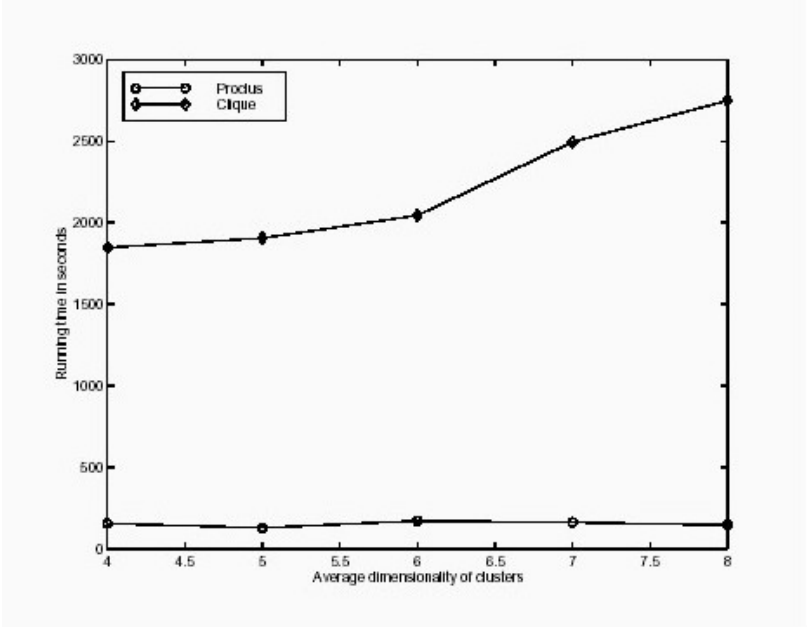
Projected Clustering

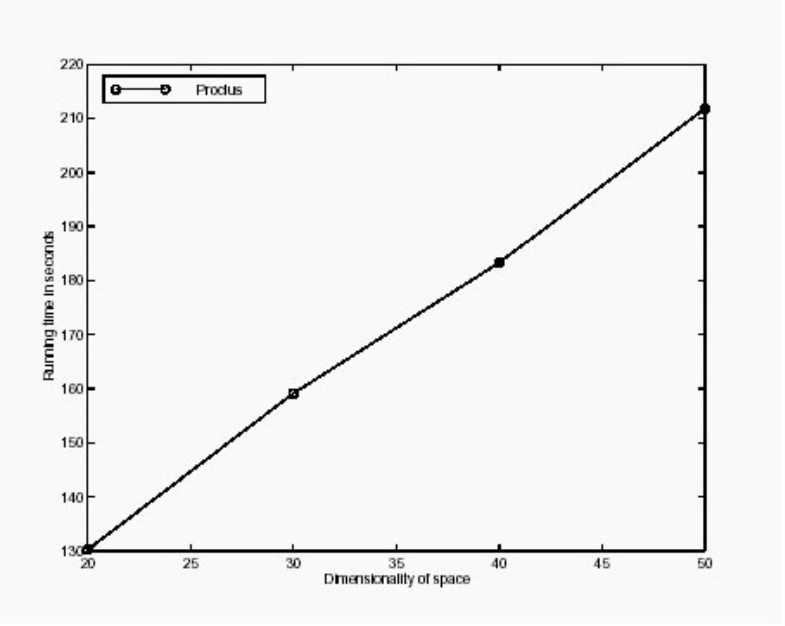
Refinement Phase

- One additional pass to improve clustering quality.
- Let C_i denote the set of points associated to m_i at the end of the iteration phase.
- Measure the average distance $X_{i,j}$ from m_i along dimension j in C_i (instead of L_i).
- For each medoid m_i, determine a new set of dimensions D_i applying the same method as in the iteration phase.
- Assign points to the closest (w.r.t. D_i) medoid m_i.
- Points that are outside of the *sphere of influence* $\Delta_i = \min_{j \neq i} dist_{D_i}(m_i, m_j)$ of all medoids are added to the set O of *outliers*.

Projected Clustering

Experimental Evaluation





Runtime complexity of PROCLUS

Linear in n , linear in d, linear in (average) dimensionality of clusters.

Projected Clustering

Discussion

Pros

Cons

Semi-supervised Clustering

Constraint-based Clustering

- Clustering with obstacle objects

  When clustering geographical data, need to take into account physical obstacles such as rivers or mountains.

  Cluster representatives must be visible from cluster elements.

- Clustering with user-provided constraints

  Users sometimes want to impose certain constraints on clusters, e.g. a minimum number of cluster elements or a minimum average salary of cluster elements.

  Two step method

  1) Find initial solution satisfying all user-provided constraints.

  2) Iteratively improve solution by moving single object to another cluster.

- Semi-supervised clustering

  → Discussed in the following section.

Semi-Supervised Clustering

Introduction

- Clustering is un-supervised learning.
- But often some constraints are available from background knowledge.
- In particular, sometimes class (cluster) labels are known for *some* of the objects.
- The resulting constraints may not all be simultaeneously satisfiable and are considered as soft (not hard) constraints.
- A *semi-supervised clustering* algorithm discovers a clustering that respects the given class label constraints as much as possible.
- Constraints in the form of *must-links* (two objects should belong to the same cluster) and *cannot-links* (two objects should not belong to the same cluster).

Semi-Supervised Clustering

A Probabilistic Framework [Basu, Bilenko & Mooney 2004]

- Based on Hidden Markov Random Fields (HMRFs).
- *Hidden* field L of n random variables whose values are unobservable, values are from {1,. . ., K}:

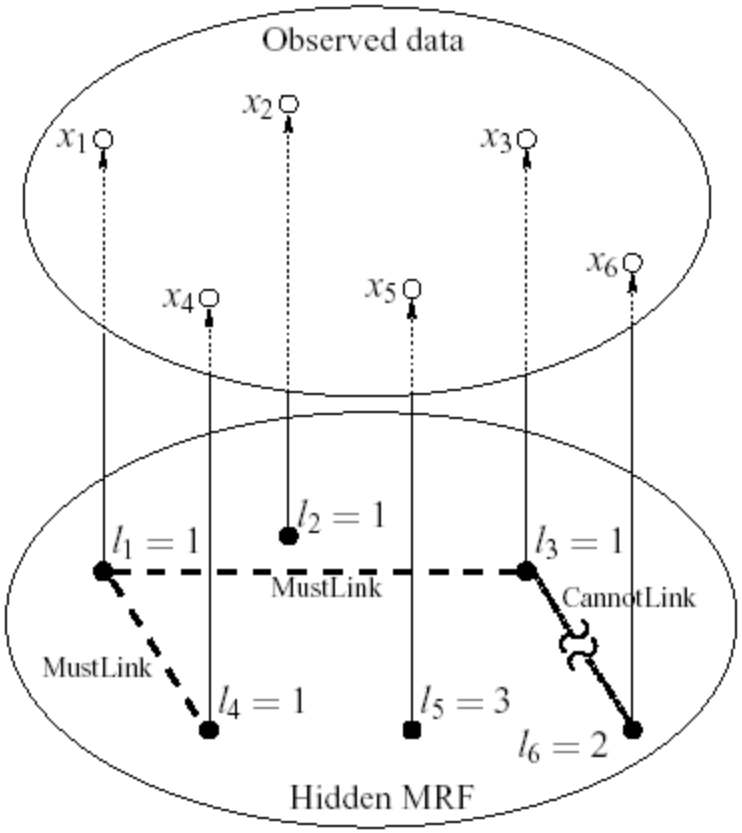
$$L = \{l_i\}_{i=1}^{n},\ X = \{x_i\}_{i=1}^{n}$$

- *Observable* set of n random variables (X):
  every x_i is generated from a conditional probability distribution determined by the hidden variables L, i.e.

$$\Pr(X \mid L) = \prod_{x_i \in X} \Pr(x_i \mid l_i)$$

Semi-Supervised Clustering

Example HMRF with Constraints



Observed variables:
data points

K = 3

Hidden variables:
cluster labels

Semi-Supervised Clustering

Properties

- Markov property

$$\forall i : \Pr(l_i \mid L - \{l_i\}) = \Pr(l_i \mid \{l_j \mid l_j \in N_i\})$$

N_i: neighborhood of l_i, i.e. variables connected to l_i via must or cannot-link

⇒ Labels depend only on labels of neighboring variables.

- Probability of a label configuration L

$$\Pr(L) = \frac{1}{Z_1} \exp(-V(L)) = \frac{1}{Z_1} \exp(-\sum_{N_i \in N} V_{N_i}(L))$$

N: set of all neighborhoods

Z_1: normalizing constant

$V(L)$: overall label configuration potential function

$V_{Ni}(L)$: potential for neighborhood N_i in configuration L

Semi-Supervised Clustering

Properties

- Since we have pairwise constraints, we consider only *pairwise potentials*:

$$\Pr(L) = \frac{1}{Z_1}\exp(-\sum_i \sum_j V(i,j)), \text{ where}$$

$$V(i,j) = \begin{cases} f_M(l_i, l_j) & \text{if } (i,j) \in M \\ f_C(l_i, l_j) & \text{if } (i,j) \in C \\ 0 & \text{otherwise} \end{cases}$$

- M: set of must-links, C: set of cannot-links.
- f_M: function that penalizes the violation of must links,
  f_C: function that penalizes the violation of cannot links.

Semi-Supervised Clustering

Properties

- $\Pr(X \mid L) = \prod_{x_i \in X} \Pr(x_i \mid l_i) = p(X, \{\mu_h\}_{h=1}^{K})$

  μ_h : representative of cluster h

- Applying Bayes theorem, we obtain

$$\Pr(L \mid X) = \left(\frac{1}{Z_2} \exp\left(-\sum_i \sum_j V(i,j)\right) \cdot p(X, \{\mu_h\}_{h=1}^{K})\right.$$

- $p(X, \{\mu_h\}_{h=1}^{K}) = \frac{1}{Z_3} \exp\left(-\sum_{x_i \in X} D(x_i, \mu_{l_i})\right)$

  $D(x_i, \mu_{l_i})$: distortion (distance) between x_i and μ_{l_i}

Semi-Supervised Clustering

Goal

- Find a label configuration L that maximizes the conditional probability (likelihood) $\Pr(L|X)$.
- There is a *trade-off* between the two factors of $\Pr(L|X)$, namely $\Pr(X|L)$ and $\mathrm{P}(L)$.
- Satisfying more label constraints increases $\mathrm{P}(L)$, but may increase the distortion and decrease $\Pr(X|L)$ (and vice versa).
- Various distortion measures can be used
  e.g., Euclidean distance, Pearson correlation, cosine similarity.
- For all these measures, there are EM type algorithms minimizing the corresponding clustering cost.

Semi-Supervised Clustering

EM Algorithm

- E-step: re-assign points to clusters based on current representatives.
- M-step: re-estimate cluster representatives based on current assignment.
- Good initialization of cluster representatives is essential.
- Assuming consistency of the label constraints, these constraints are exploited to generate λ neighborhoods with representatives.
- If $\lambda < k$, then determine k-λ additional representatives by random perturbations of the global centroid of X.
- If $\lambda > k$, then k of the given representatives are selected that are maximally separated from each other (w.r.t. D).

Semi-Supervised Clustering

Semi-Supervised Projected Clustering [Yip et al 2005]

- Supervision in the form of labeled objects, i.e. (object, class label) pairs, and labeled dimensions, i.e. (class label, dimension) pairs.
- Input parameter is *k* (number of clusters).
- No parameter specifying the average number of dimensions (parameter *l* in PROCLUS).
- Objective function essentially measures the average variance over all clusters and dimensions.
- Algorithm similar to *k*-medoid.
- Initialization exploits user-provided labels.
- Can effectively find very low-dimensional projected clusters.

One Minute Survey

What clustering algorithm would you recommend for

- transaction data,
- GPS data,
- tweets,
- proteins?

→ Discuss with your neighbor.

→ Share your ideas.

References

[Aggarwal et al 1999]

Charu C. Aggarwal, C. Procopiuc, J.L. Wolf, Philip S. Yu, J.S. Park: Fast Algorithms for Projected Clustering, SIGMOD 1999.

[Agrawal et al 1998]

Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopoulos, Prabhakar Raghavan: Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, SIGMOD 1998.

[Ankerst, Breunig, Kriegel & Sander 1999]

Mihael Ankerst, Markus Breunig, Hans=Peter Kriegel, Joerg Sander: OPTICS: Ordering Points To Identify the Clustering Structure, SIGMOD 1999.

[Basu, Bilenko & Mooney 2004]

Sugatu Basu, Mikhail Bilenko, Raymond Mooney: A probabilistic framework for semi-supervised clustering, KDD 2004.

[Lee & Seung 2001]

Algorithms for Non-negative Matrix Factorization, NIPS 2001.

[Dempster, Laird & Rubin 1977]

A. P. Dempster; N. M. Laird; D. B. Rubin. Journal of the Royal Statistical Society. Series B (Methodological), Vol. 39, No. 1.

References

[Ester, Kriegel, Sander & Xu 1996]

Martin Ester, Hans-Peter Kriegel, Joerg Sander, Xiaowei Xu: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, KDD 1996.

[Fayyad, Reina & Bradley 1998]

Usama Fayyad, Cory Reina, Paul Bradley: Initialization of Iterative Refinement Clustering Algorithms, KDD 1998.

[Forgy 1965]

E.W. Forgy: "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". Biometrics 21, 1965.

[Guha, Rastogi & Shim 1998]

Sudipto Guha, Rajeev Rastogi, Kyuseok Shim: CURE: An Efficient Clustering Algorithm for Large Databases, SIGMOD 1998.

[Jain & Dubes 1988]

Anil K. Jain, Richard C. Dubes: Algorithms for Clustering Data. Prentice-Hall 1988.

[Kaufman and Rousseeuw 1990]

L. Kaufman, Peter Rousseeuw: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley 1990.

References

[MacQueen 1967]

J. B. MacQueen: Some Methods for classification and Analysis of Multivariate Observations, 5-th Berkeley Symposium on Mathematical Statistics and Probabilit*y,* 1967.

[Ng & Han 1994]

Raymond T. Ng, Jiawei Han: Efficient and Effective Clustering Methods for Spatial Data Mining, VLDB 1994.

[Schubert et al 2017]

Erich Schubert, Jörg Sander, Martin Ester, Hans-Peter Kriegel, Xiaowei Xu: DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. ACM Trans. Database Syst. 42(3): 19:1-19:21 (2017)

[Wang, Wang, Yang &Yu 2002]

Haixun Wang, Wei Wang, Jiong Yang, Philip Yu: Clustering by pattern similarity in large data sets, SIGMOD 2002.

[Yip et al 2005]

Kevin Y. Yip, David W. Cheung, Michael K. Ng: On Discovery of Extremely Low-Dimensional Clusters using Semi-Supervised Projected Clustering, ICDE 2005.