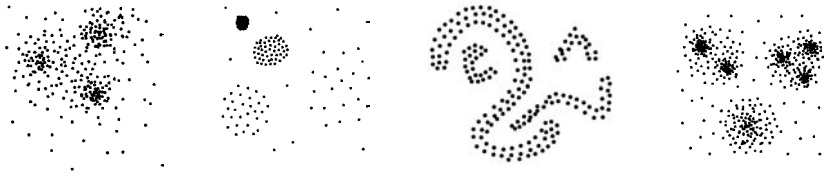
4. Cluster and Outlier Analysis

Contents of this Chapter

4.1 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

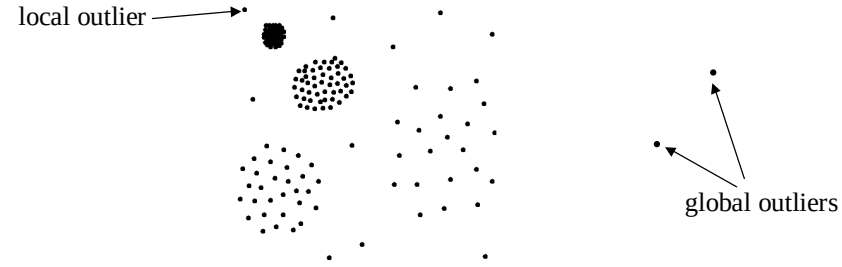


clusters of different sizes, shapes, densities
hierarchical clusters
disjoint / overlapping clusters

4.1 Introduction

Goal of Outlier Analysis

- Identification of objects (*outliers*) in the dataset which are
  significantly different from the rest of the dataset (*global outliers*)
  or significantly different from their neighbors in the dataset (*local outliers*)



outliers do not belong to any of the clusters

4.1 Introduction

Clustering as Optimization Problem

Definition
- dataset D, $|D| = n$
- *clustering C of D*:

$$C = \{C_1, ..., C_k\}$$

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

Goal

find *clustering* that *best fits* the given training data

Search Space

space of all clusterings

size is $O(2^n)$

local optimization methods (greedy)

4.1 Introduction

Clustering as Optimization Problem

Steps

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

4.1 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)
$dist(o_1, o_3) \leq dist(o_1, o_2) + dist(o_2, o_3)$.

4.1 Distance Functions

Distance Functions for Numeric 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]

4.1 Distance Functions

Other Distance Functions

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

- for text documents D (vectors of frequencies of terms of T)

  $d = \{f(t_i, D) | t_i \in T\}$ $f(t_i, D)$: frequency of term t_i in document D

  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

4.1 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)

4.1 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]>

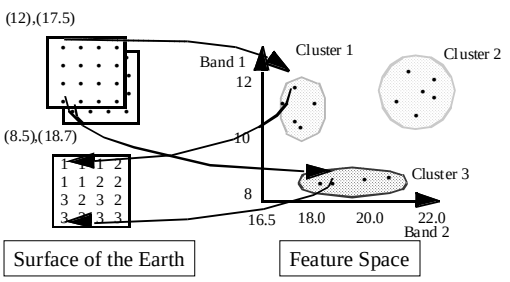
⇨ which entries form a session?

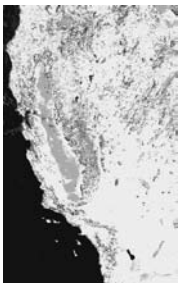
Distance Function for Sessions

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

4.1 Typical Clustering Applications

Generating Thematic Maps from Satellite Images





Assumption

Different land usages exhibit different / characteristic properties of reflection and emission

4.1 Types of Clustering Methods

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

Hierarchical Methods
- Parameters: distance function for objects and for clusters
- determines a hierarchy of clusterings, merges always the most similar clusters

Density-Based Methods
- Parameters: minimum density within a cluster, distance function
- extends cluster by neighboring objects as long as the density is large enough

Other Clustering Methods
- Fuzzy Clustering
- Graph-based Methods
- Neural Networks

4.2 Partitioning Methods

Basics

Goal

a (disjoint) partitioning into *k* clusters with minimal costs

Local optimization method

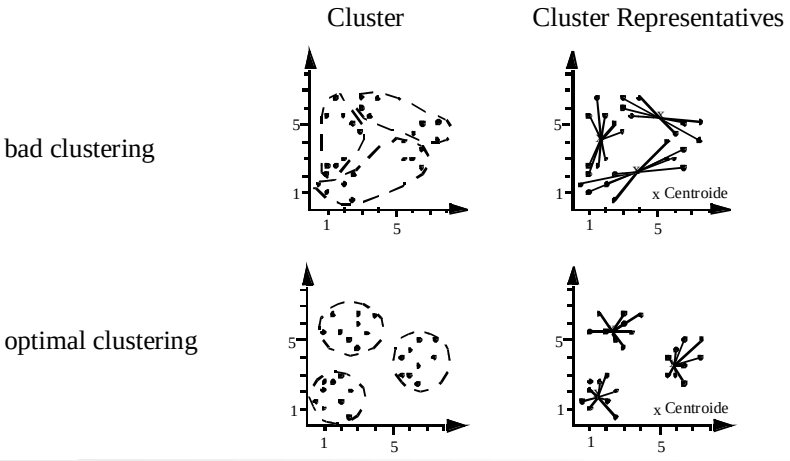
- choose *k* initial cluster representatives
- optimize these representatives iteratively
- assign each object to its most similar cluster representative

Types of cluster representatives

- Mean of a cluster (*construction of central points*)
- Median of a cluster (*selection of representative points*)
- Probability density function of a cluster (*expectation maximization*)

4.2 Construction of Central Points

Example



4.2 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 costs* (compactness) *of a clusters C*

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

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

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

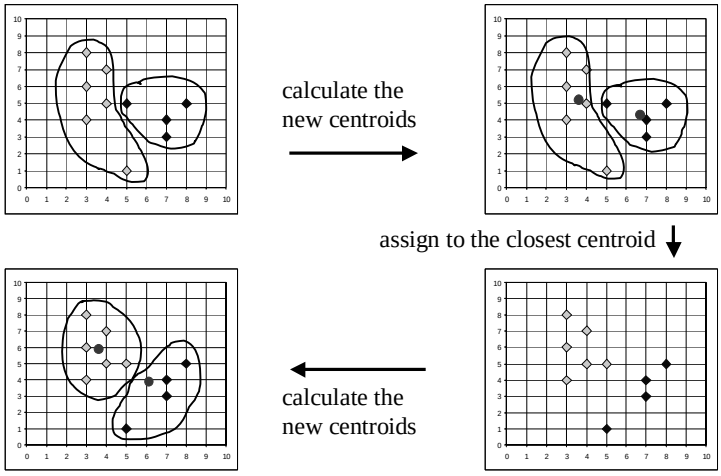
4.2 Construction of Central Points

Algorithm

```
ClusteringByVarianceMinimization(dataset D, integer k)
  create an „initial“ partitioning of dataset D into k
   clusters;
  calculate the set C'={C1, ..., Ck} 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;
```

4.2 Construction of Central Points

Example



4.2 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

4.2 Construction of Central Points

Discussion

+ Effiziency
  Runtime: O(*n*) for one iteration,
  number of iterations is typically small (~ 5 - 10).
+ simple implementation

⇒ *K*-means is the most popular partitioning clustering method

- sensitivity to noise and outliers
  all objects influence the calculation of the centroid
- all clusters have a convex shape
- the number *k* of clusters is often hard to determine
- highly dependent from the initial partitioning
  clustering result as well as runtime

4.2 Selection of Representative Points

Basics [Kaufman & Rousseeuw 1990]

- Assumes only a distance function for pairs of objects
- *Medoid*: a representative element of the cluster (representative point)
- *Measure for the costs* (compactness) *of a clusters C*

$$TD(C) = \sum_{p \in C} dist(p, m_C)$$

- *Measure for the costs* (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 of exhaustive search $O(n^k)$

4.2 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 costs 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

4.2 Selection of Representative Points

Algorithm PAM

```
PAM(dataset D, integer k, float dist)
  initialize the k medoids;
  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;
  return best k medoids;
```

4.2 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;
```


4.2 Selection of Representative Points

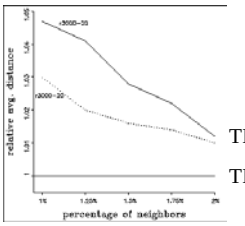
Comparison of PAM and CLARANS

Runtime complexities

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

Experimental evaluation

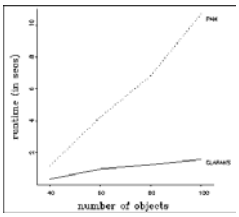
Quality



TD(CLARANS)

TD(PAM)

Runtime



4.2 Expectation Maximization

Basics [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 clusters C
  - mean μ_C of all cluster points
  - d x d covariance matrix Σ_C for the points of cluster C
- *probability density function* of *cluster C*

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

4.2 Expectation Maximization

Basics

- *probability density function* of *clustering* $M = \{C_1, . . ., C_k\}$

$$P(x) = \sum_{i=1}^{k} W_i \cdot P(x|C_i)$$

with W_i percentage of points of D in C_i

- assignment of points to clusters

$$P(C_i|x) = W_i \cdot \frac{P(x|C_i)}{P(x)}$$

point belongs to several clusters with different probabilities

- measure of clustering quality (likelihood)

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

⇒ the larger the value of E, the higher the probability of dataset D

$E(M)$ is to be maximized

4.2 Expectation Maximization

Algorithm

```
ClusteringByExpectationMaximization
      (dataset D, integer k)
  create an „initial“ clustering M' = (C1', ..., Ck');
  repeat // re-assignment
    calculate P(x|Ci), P(x) and P(Ci|x) for each
     object x of D and each cluster Ci;
    // re-calculation of clustering
    calculate a new clustering M ={C1, ..., Ck} by
     re-calculating Wi, μC and ΣC for each i;
    M' := M;
  until |E(M) - E(M')| < ε;
  return M;
```


4.2 Expectation Maximization

Discussion

- converges to a (possibly *local*) minimum
- runtime complexity:

  O($n * k * \#iterations$)

  # iterations is typically large

- clustering result and runtime strongly depend on
  - initial clustering
  - „correct“ choice of parameter *k*
- modification for determining *k disjoint* clusters:

  assign each object x only to cluster C_i with maximum $P(C_i|x)$

4.2 Choice of Initial Clusterings

Idea

- in general, clustering of a small sample yields good initial clusters
- but some samples may have a significantly different distribution

Method [Fayyad, Reina & Bradley 1998]

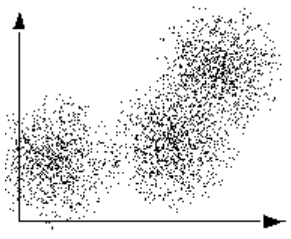
- draw independently *m* different samples
- cluster each of these samples

  ⇒ *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 one with the highest clustering quality as initial clustering for the whole dataset

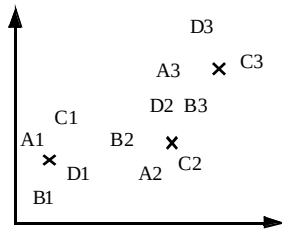
4.2 Choice of Initial Clusterings

Example



whole dataset

$k = 3$



DB

from $m = 4$ samples

× *true cluster means*

4.2 Choice of Parameter *k*

Method

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

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 decreases monotonically with increasing *k*

4.2 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, . . .

4.3 Hierarchical Methods

Basics

Goal

construction of a hierarchy of clusters (*dendrogram*)
merging clusters with minimum distance

Dendrogram

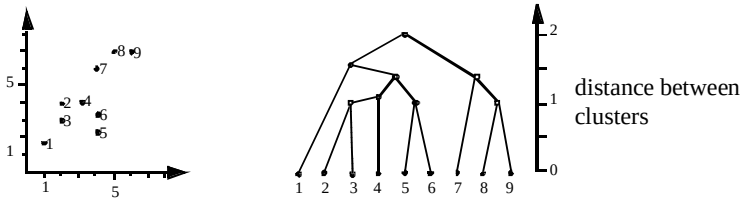
a tree of nodes representing clusters, satisfying the following properties:

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

4.3 Hierarchical Methods

Basics

Example dendrogram



Types of hierarchical methods

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

4.3 Single-Link and Variants

Algorithm Single-Link [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.

4.3 Single-Link and Variants

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

4.3 Single-Link and Variants

Discussion

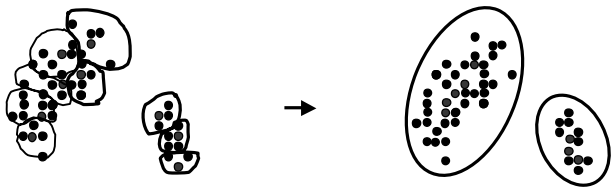
+ does not require knowledge of the number *k* of clusters
+ finds not only a „flat“ clustering, but a hierarchy of clusters (dendrogram)
+ a single clustering can be obtained from the dendrogram
  (e.g., by performing a horizontal cut)

- decisions (merges/splits) cannot be undone
- sensitive to noise (Single-Link)
  a „line“ of objects can connect two clusters
- inefficient
  runtime complexity at least $O(n^2)$ for *n* objects

4.3 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

4.4 Density-Based Clustering

Basics

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

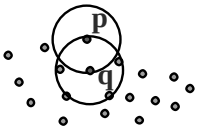
Strenghts of density-based clustering

- clusters of arbitrary shape
- robust to noise
- efficiency

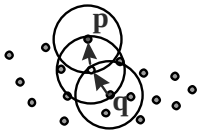
4.4 Density-Based Clustering

Basics [Ester, Kriegel, Sander & Xu 1996]

- object o ∈ *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.

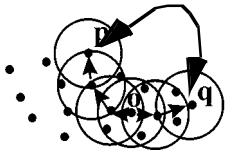


border object: no core object,
but density-reachable from other object (p)

4.4 Density-Based Clustering

Basics

- objects *p* and *q* are *density-connected*: both are density-reachable from a third object *o*.



- *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*.

4.4 Density-Based Clustering

Basics

- 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*$\}$.

4.4 Density-Based Clustering

Algorithm DBSCAN

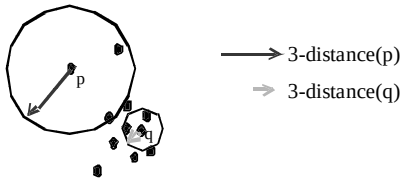
```
DBSCAN(dataset D, float ε, integer MinPts)
  // all objects are initially unclassified,
  // o.ClId = UNCLASSIFIED for all o ∈ D

  ClusterId := nextId(NOISE);
  for i from 1 to |D| do
    object := D.get(i);
    if Objekt.ClId = UNCLASSIFIED then
         if ExpandCluster(D, object, ClusterId, ε,
           MinPts)
         then ClusterId:=nextId(ClusterId);
```


4.4 Density-Based Clustering

Choice of Parameters

- cluster: density above the „minimum density“ defined by ε and *MinPts*
- wanted: the cluster with the lowest density
- heuristic method: consider the distances to the *k*-nearest neighbors

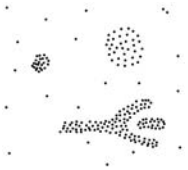


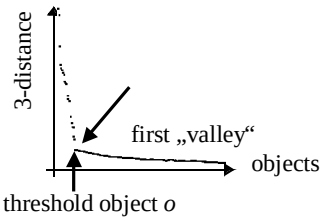
- function *k-distance*: distance of an object to its *k*-nearest neighbor
- *k-distance-diagram*: *k*-distances in descending order

4.4 Density-Based Clustering

Choice of Parameters

Example





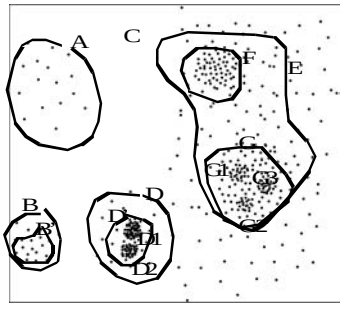
Heuristic Method

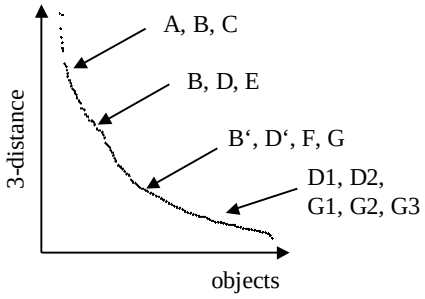
- User specifies a value for *k* (Default is *k* = 2**d - 1*), *MinPts* := *k*+1.
- System calculates the *k*-distance-diagram for the dataset and visualizes it.
- User chooses a threshold object from the *k*-distance-diagram, ε := *k*-distance(*o*).

4.4 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

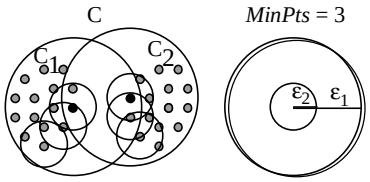




4.4 Hierarchical Density-Based Clustering

Basics [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

- does not generate a dendrogramm, but a graphical visualization of the hierarchical cluster structure

4.4 Hierarchical Density-Based Clustering

Basics

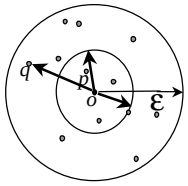
Core distance of object *p* w.r.t. ε and *MinPts*

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

Reachability distance of object *p* relative zu 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*)

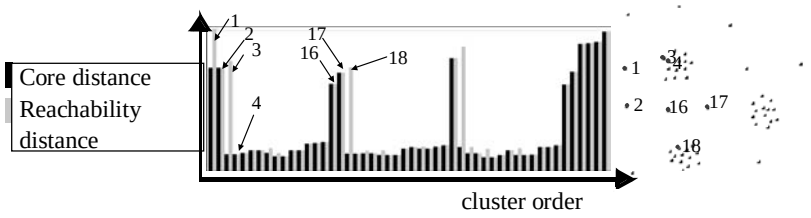
→ *Reachability distance*(*p*,*o*)

→ *Reachability distance*(*q*,*o*)

4.4 Hierarchical Density-Based Clustering

Cluster Order

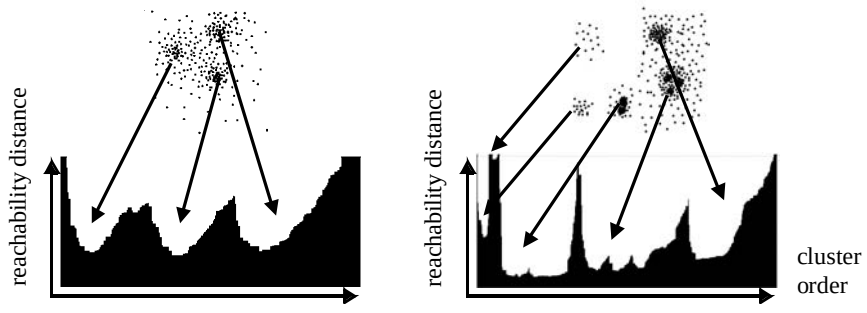
- OPTICS does not directly return a (hierarchichal) clustering, but orders the objects according to a „cluster order“ w.r.t. ε and *MinPts*
- *cluster order* w.r.t. ε and *MinPts*
  - start with an arbitrary object
  - visit the object that has the minimum reachability distance from the *set of already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



4.4 Hierarchical Density-Based Clustering

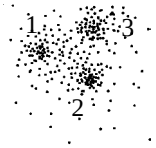
Reachability Diagram

- 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

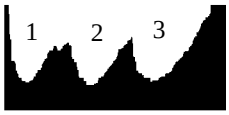


4.4 Hierarchical Density-Based Clustering

Sensitivity of Parameters

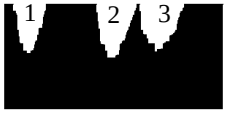


MinPts = 10, ε = 10



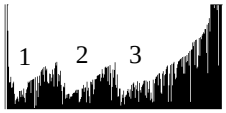
"optimum" parameters

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"

4.4 Hierarchical Density-Based Clustering

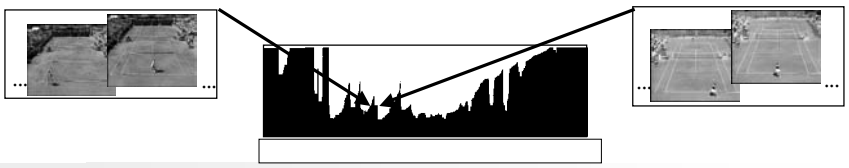
Heuristics for Setting theParameters

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



4.4 Hierarchical Density-Based Clustering

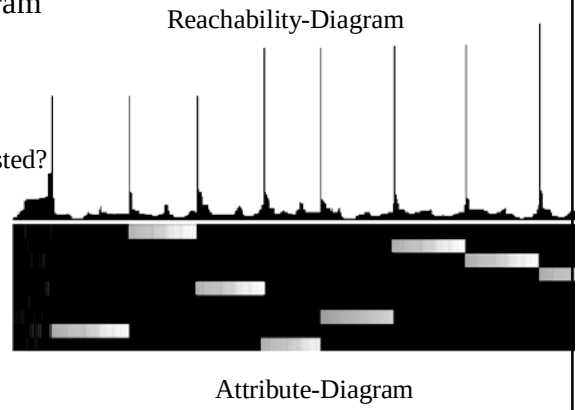
Manual Cluster Analysis

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?

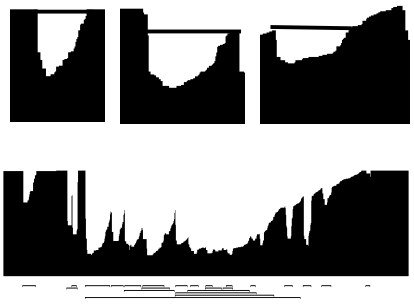


4.4 Hierarchical Density-Based Clustering

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*)

4.5 Database Techniques for Scalable Clustering

Goal

So far

- small datasets
- in main memory

Now

- very large datasets which do not fit into main memory
- data on secondary storage (*pages*)
  random access orders of magnitude more expensive than in main memory

scalable clustering algorithms

4.5 Database Techniques for Scalable Clustering

Use of Spatial Index Structures or Related Techniques

- index structures obtain a coarse pre-clustering (*micro-clusters*)
  neighboring objects are stored on the same / a neighboring disk block
- index structures are efficient to construct
  based on simple heuristics
- fast access methods for similarity queries
  e.g. region queries and *k*-nearest-neighbor queries

4.5 Region Queries for Density-Based Clustering

- basic operation for DBSCAN and OPTICS:
  retrieval of ε-neighborhood for a database object *o*
- efficient support of such region queries by spatial index structures such as

  R-tree, X-tree, M-tree, . . .
- runtime complexities for DBSCAN and OPTICS:

	single range query	whole algorithm
without index	$O(n)$	$O(n^2)$
with index	$O(\log n)$	$O(n \log n)$
with random access	$O(1)$	$O(n)$

spatial index structures degenerate for very high-dimensional data

4.5 Index-Based Sampling

Method [Ester, Kriegel & Xu 1995]

- build an R-tree (often given)
- select sample objects from the data pages of the R-tree
- apply the clustering method to the set of sample objects (in memory)
- transfer the clustering to the whole database (one DB scan)

data pages
of an R-tree

sample has similar
distribution as DB

4.5 Index-Based Sampling

Transfer the Clustering to the whole Database

- For *k*-means- and *k*-medoid-methods:

  apply the cluster representatives to the whole database (centroids, medoids)
- For density-based methods:

  generate a representation for each cluster (e.g. bounding box)

  assign each object to closest cluster (representation)
- For hierarchichal methods:

  generation of a hierarchical representation (dendrogram or reachability-diagram) from the sample is difficult

4.5 Index-Based Sampling

Choice of Sample Objects

How many objects per data page?

- depends on clustering method
- depends on the data distribution
- e.g. for CLARANS: one object per data page

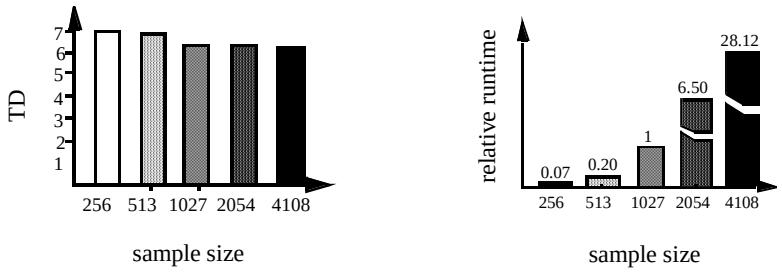
good trade-off between clustering quality and runtime

Which objects to choose?

- simple heuristics: choose the „central“ object(s) of the data page

4.5 Index-Based Sampling

Experimental Evaluation for CLARANS



- runtime of CLARANS is approximately O(n^2)
- clustering quality stabilizes for more than 1024 sample objects

4.5 Data Compression for Pre-Clustering

Basics [Zhang, Ramakrishnan & Linvy 1996]

Method

- determine compact summaries of "micro-clusters" (Clustering Features)
- hierarchical organization of clustering features
  in a balanced tree (CF-tree)
- apply any clustering algorithm, e.g. CLARANS
  to the leaf entries (micro-clusters) of the CF-tree

CF-tree

- compact, hierarchichal representation of the database
- conserves the "cluster structure"

4.5 Data Compression for Pre-Clustering

Basics

Clustering Feature of a set C of points X_i: $CF = (N, LS, SS)$

$N = |C|$ number of points in C

$LS = \sum_{i=1}^{N} X_i$ linear sum of the N points

$SS = \sum_{i=1}^{N} X_i^2$ square sum of the N points

CFs sufficient to calculate

- centroid
- measures of compactness
- and distance functions for clusters

4.5 Data Compression for Pre-Clustering

Basics

Additivity Theorem

CFs of two disjoint clusters C_1 and C_2 are additive:

$$CF(C_1 \cup C_2) = CF(C_1) + CF(C_2) = (N_1 + N_2, LS_1 + LS_2, QS_1 + QS_2)$$

i.e. CFs can be incrementally calculated

Definition

A *CF-tree* is a height-balanced tree for the storage of CFs.

4.5 Data Compression for Pre-Clustering

Basics

Properties of a CF-tree

- Each innner node contains at most B entries [CF_i, $child_i$] and CF_i is the CF of the subtree of $child_i$.
- A leaf node contains at most L entries [CF_i].
- Each leaf node has two pointers *prev* and *next*.
- The diameter of each entry in a leaf node (micro-cluster) does not exceed T.

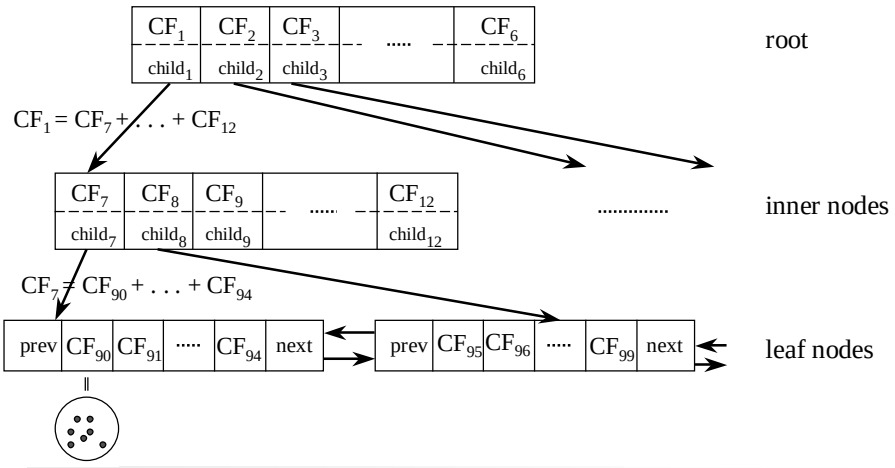
Construction of a CF-tree

- Transform an object (point) p into clustering feature CF_p=(1, p, p^2).
- Insert CF_p into closest leaf of CF-tree (similar to B^+-tree insertions).
- If diameter threshold T is violated, split the leaf node.

4.5 Data Compression for Pre-Clustering

Example

B = 7, L = 5



4.5 Data Compression for Pre-Clustering

BIRCH

Phase 1

- one scan of the whole database
- construct a CF-tree B_1 w.r.t. T_1 by successive insertions of all data objects

Phase 2

- if CF-tree B_1 is too large, choose $T_2 > T_1$
- construct a CF-tree B_2 w.r.t. T_2 by inserting all CFs from the leaves of B_1

Phase 3

- apply any clustering algorithm to the CFs (micro-clusters) of the leaf nodes of the resulting CF-tree (instead to all database objects)
- clustering algorithm may have to be adapted for CFs

4.5 Data Compression for Pre-Clustering

Discussion

+ CF-tree size / compression factor is a user parameter
+ efficiency

  construction of secondary storage CF-tree: O(n log n) [page accesses]

  construction of main memory CF-tree : O(n) [page accesses]

  additionally: cost of clustering algorithm

- only for numeric data
  Euclidean vector space
- result depends on the order of data objects

4.6 Special Requirements and Methods

Overview

- categoric attributes

  analogue to means as cluster representatives (see assignment)

- spatially extended objects

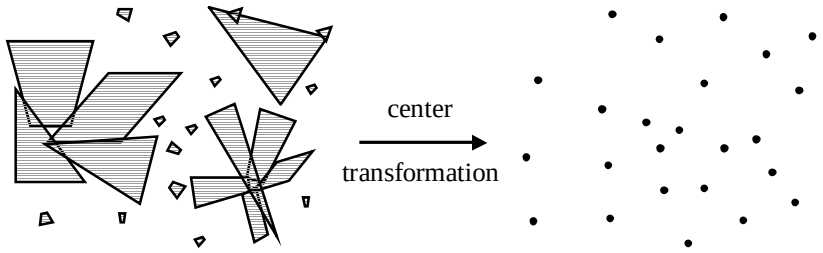
  generalised density-based clustering

- clusters only in subspaces of the data space

  subspace clustering

4.6 Clustering of Spatially Extended Objects

Motivation



center transformation destroys cluster structure
have to consider the spatial extension of objects

4.6 Clustering of Spatially Extended Objects

Generalized Density-Based Clustering

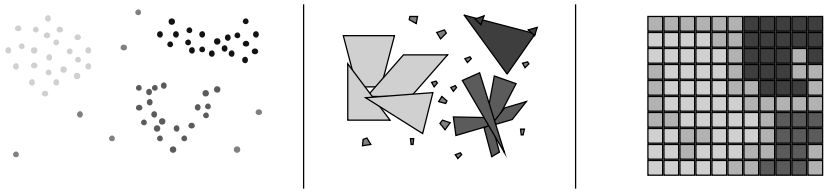
[Sander, Ester, Kriegel & Xu 1998]

„ε-neighborhood contains at least *MinPts* objects“

$dist(o, p) \leq \varepsilon$	$\lvert N_\varepsilon(o)\rvert \geq MinPts$
generalize ↓	↓ generalize
NPred(*o*,*p*) reflexive symmetric	*MinWeight*($N_\varepsilon(o)$) monotone
$N_{NPred}(o) = \{p \in D \mid NPred(o, p)\}$	*MinWeight*($N_{NPred}(o)$)

4.6 Clustering of Spatially Extended Objects

Examples



NPred		
dist(p,q) ≤ ε	intersect(p,q)	neighboring cell with same color
MinWeight		
cardinality(. . .) ≥ MinPoints	sum of areas ≥ 5 % of total area	true

4.6 Clustering of Spatially Extended Objects

Algorithm GDBSCAN

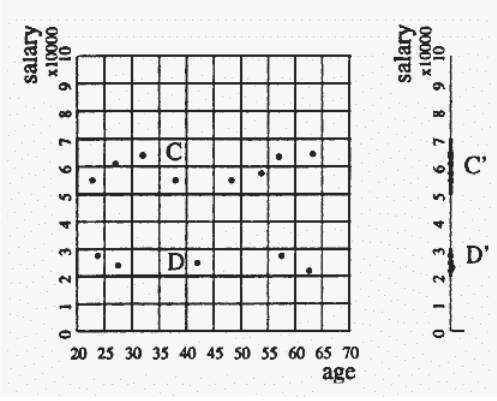
- same algorithmic scheme as DBSCAN
- N_{NPred}-query instead of N_ε-query
- evaluate *MinWeight*–predicate

  instead of condition $|N_\varepsilon| \geq MinPts$
- runtime complexity O(n log n)

  if N_{NPred}-query efficiently supported, i.e. O(log n)

4.6 Subspace Clustering

Motivation



clusters only in
1-dimensional subspace
„salary“

4.6 Subspace Clustering

CLIQUE [Agrawal, Gehrke, Gunopulos & Raghavan 1998]

1. identification of subspaces with clusters
2. identification of clusters
3. generation of cluster descriptions

- *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)

4.6 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)

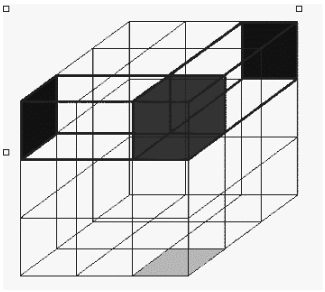
  start with the empty set

  add one more dimension at a time
- foundation of this algorithm: *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)

4.6 Subspace Clustering

Example



2-dim. dense regions

3-dim. candidate region

2-dim. region to be tested

- runtime complexity of greedy algorithm $O(\varsigma^k + n \cdot k)$
  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

4.6 Subspace Clustering

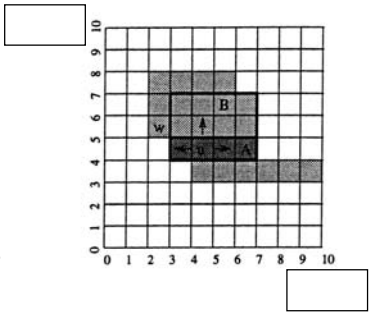
Identification of Clusters

- 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*

4.6 Subspace Clustering

Generation of Cluster Descriptions

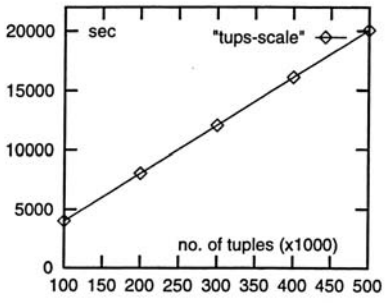
- given: a cluster, i.e. a set of connected dense base regions
- task: find optimal cover of this cluster
  by a set of hyperrectangles
- standard methods
  infeasible for large values of *d*
  the problem is NP-hard
- heuristic method
  1. cover the cluster by maximal regions
  2. remove redundant regions



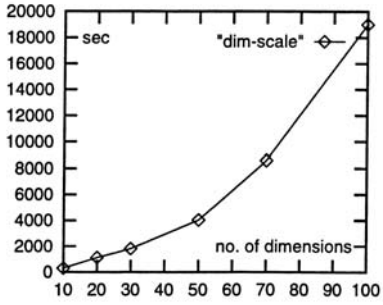
4.6 Subspace Clustering

Experimental Evaluation

runtime



runtime



runtime complexity of CLIQUE
linear in *n* , superlinear in *d*

4.6 Subspace Clustering

Discussion

+ automatic detection of subspaces with clusters
+ no assumptions on the data distribution
+ independent from the order of the data objects
+ scalable w.r.t. the number *n* of data objects

- accuracy crucially depends on parameter ξ
- needs a heuristics to reduce the search space (all subsets of dimensions)

⇒ method is not complete

4.7 Outlier Detection

Overview

Definition

Outliers: objects *significantly dissimilar* from the remainder of the data

Applications

- Credit card fraud detection
- Telecom fraud detection
- Medical analysis

Problem

- Find top *k* outlier points

4.7 Outlier Detection

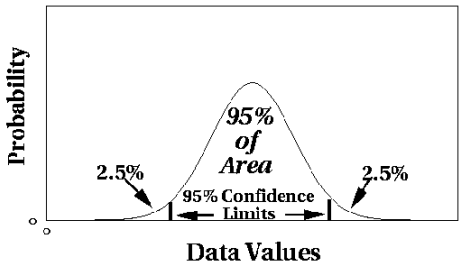
Statistical Approach

Assumption

Statistical model that generates data set (e.g. normal distribution)

Use tests depending on

- data distribution
- distribution parameter (e.g., mean, variance)
- number of expected outliers



Drawbacks

- most tests are for single attribute
- data distribution may not be known

4.7 Outlier Detection

Distance-Based Approach

Idea

outlier analysis without knowing data distribution

Definition

DB(p, t)-outlier:

object o in a dataset D such that at least a fraction p of the objects in D has a distance greater than t from o

Algorithms for mining distance-based outliers

- Index-based algorithm
- Nested-loop algorithm
- Cell-based algorithm

4.7 Outlier Detection

Deviation-Based Approach

Idea

- Identifies outliers by examining the main characteristics of objects in a group
- Objects that "deviate" from this description are considered outliers

Sequential exception technique

- simulates the way in which humans can distinguish unusual objects from among a series of supposedly like objects

OLAP data cube technique

- uses data cubes to identify regions of anomalies in large multidimensional data
- Example: city with significantly higher sales increase than its region