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

Skript zur Vorlesung

Knowledge Discovery in Databases

im Wintersemester 2009/2010

Kapitel 5: Clustering

Skript © 2003 Johannes Aßfalg, Christian Böhm, Karsten Borgwardt, Martin Ester, Eshref Januzaj, Karin Kailing, Peer Kröger, Jörg Sander und Matthias Schubert

http://www.dbs.ifi.lmu.de/Lehre/KDD

5 Clustering

Inhalt dieses Kapitels

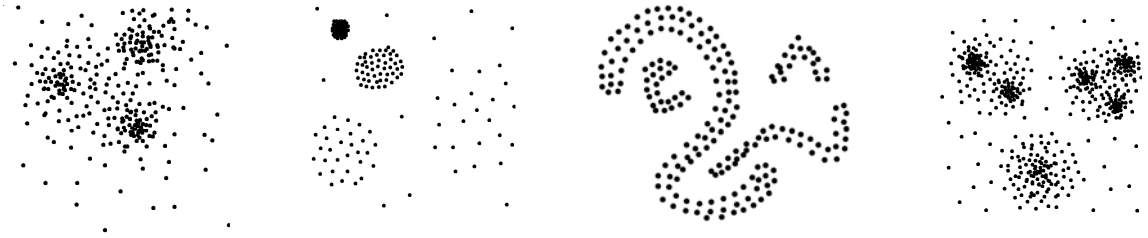
5.1 Einleitung
Ziel des Clustering, Anwendungen, Typen von Clustering-Algorithmen
5.2 Partitionierende Verfahren
k-means, k-medoid, EM, Initialisierung und Parameterwahl
5.3 Dichtebasierte Verfahren
Grundlagen, DBSCAN, SNN
5.4 Hierarchische Verfahren
Single-Link und Varianten, dichtebasiertes hierarchisches Clustering
5.5 Besondere Anwendungen
k-modes, verallgemeinertes dichte-basiertes Clustering

Ziel des Clustering

Identifikation einer endlichen Menge von Kategorien, Klassen oder Gruppen
(*Cluster*) in den Daten

Objekte im *gleichen* Cluster sollen möglichst ähnlich sein

Objekte aus *verschiedenen* Clustern sollen möglichst unähnlich zueinander sein



Cluster unterschiedlicher Größe, Form und Dichte
hierarchische Cluster
=> unterschiedliche Clustering-Algorithmen

Typische Anwendungen

- Kundensegmentierung
  Clustering der Kundentransaktionen
- Bestimmung von Benutzergruppen auf dem Web
  Clustering der Web-Logs
- Strukturierung von großen Mengen von Textdokumenten
  Hierarchisches Clustering der Textdokumente
- Erstellung von thematischen Karten aus Satellitenbildern
  Clustering der aus den Rasterbildern gewonnenen Featurevektoren

Typen von Clustering Verfahren

- Partitionierende Verfahren
  - Parameter: Anzahl *k* der Cluster, Distanzfunktion
  - sucht ein „flaches" Clustering in *k* Cluster mit minimalen Kosten
- Hierarchische Verfahren
  - Parameter: Distanzfunktion für Punkte und für Cluster
  - bestimmt Hierarchie von Clustern, mischt jeweils die ähnlichsten Cluster
- Dichtebasierte Verfahren
  - Parameter: minimale Dichte in einem Cluster, Distanzfunktion
  - erweitert Punkte um ihre Nachbarn solange Dichte groß genug
- Andere Clustering-Verfahren
  - Fuzzy Clustering, Graph-theoretische Verfahren, neuronale Netze

Grundlagen

Ziel: Partitionierung in *k* Cluster, so dass eine Kostenfunktion minimiert wird (Gütekriterium)

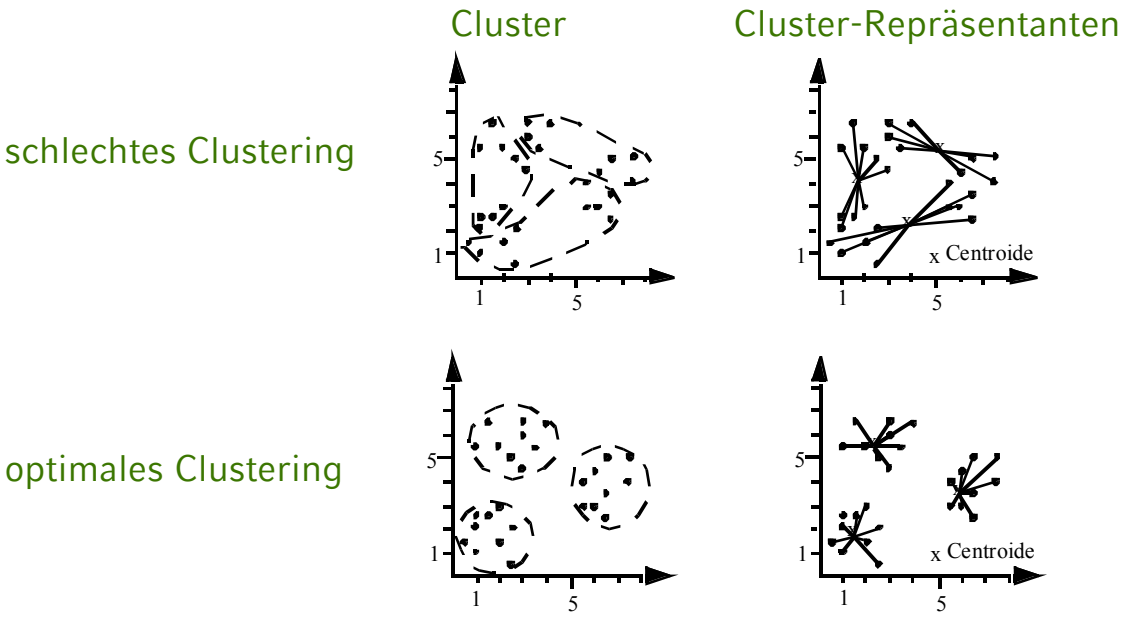
Lokal optimierendes Verfahren

- wähle *k* initiale Cluster-Repräsentanten
- optimiere diese Repräsentanten iterativ
- ordne jedes Objekt seinem ähnlichsten Repräsentanten zu

Typen von Cluster-Repräsentanten

- Mittelwert des Clusters (*Konstruktion zentraler Punkte*)
- Element des Clusters (*Auswahl repräsentativer Punkte*)
- Wahrscheinlichkeitsverteilung des Clusters (*Erwartungsmaximierung*)

Konstruktion zentraler Punkte (Beispiel)



Konstruktion zentraler Punkte (Grundbegriffe)

[Forgy 1965]

- Objekte sind Punkte $x = (x_1, ..., x_d)$ in einem euklidischen Vektorraum

  dist = euklidische Distanz (L_2-Norm)

- *Centroid* μ_C: Mittelwert aller Punkte im Cluster *C*

- *Maß für die Kosten* (Kompaktheit) *eines Clusters C*

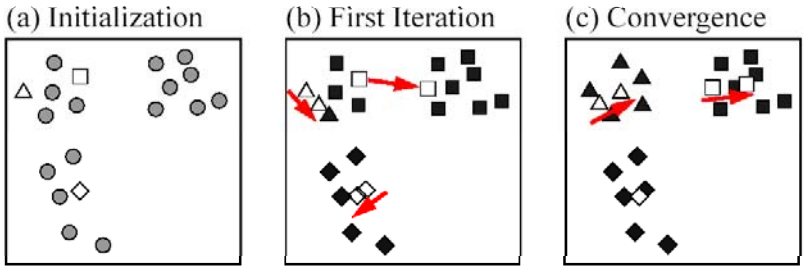
$$TD^2(C) = \sum_{p \in C} dist(p, \mu_C)^2$$

- *Maß für die Kosten* (Kompaktheit) *eines Clustering*

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

Idee des Algorithmus

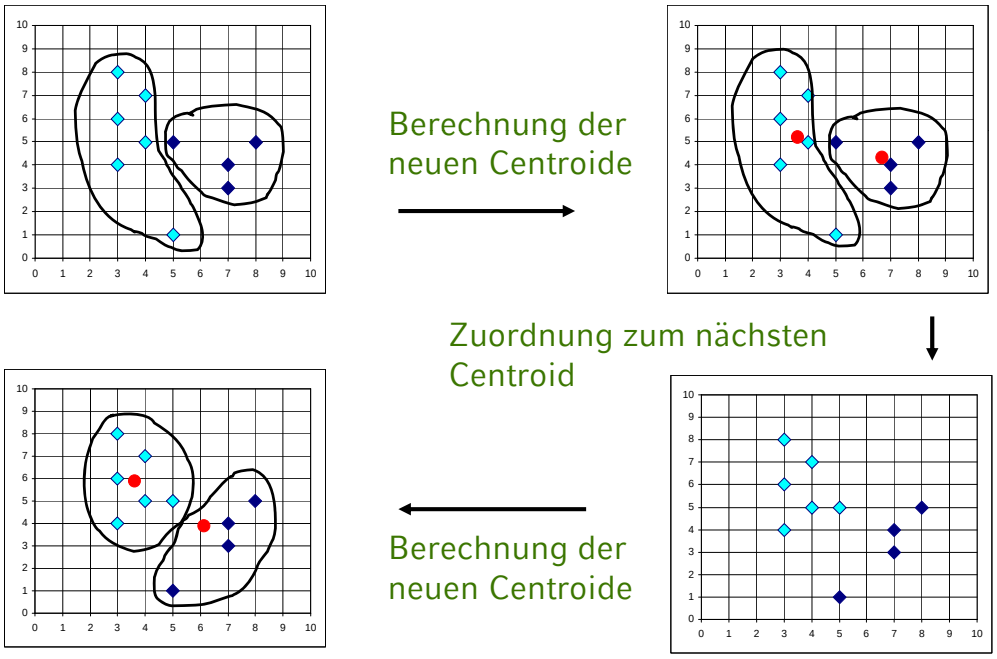
- Algorithmus startet z.B. mit zufällig gewählten Punkten als Cluster-Repräsentanten
- Der Algorithmus besteht aus zwei alternierenden Schritten:
  - Zuordnung jedes Datenpunktes zum räumlich nächsten Repräsentanten
  - Neuberechnung der Repräsentanten (Centroid der zugeordneten Punkte)
- Diese Schritte werden so lange wiederholt, bis sich keine Änderung mehr ergibt



Algorithmus

```
ClusteringDurchVarianzMinimierung(Punktmenge D, Integer k)
    Erzeuge eine „initiale“ Zerlegung der Punktmenge D in k
      Klassen;
    Berechne die Menge C'={C1, ..., Ck} der Centroide für
      die k Klassen;
    C = {};
    repeat
        C = C';
        Bilde k Klassen durch Zuordnung jedes Punktes zum
          nächstliegenden Centroid aus C;
        Berechne die Menge C'={C'1, ..., C'k} der Centroide
          für die neu bestimmten Klassen;
    until C = C';
    return C;
```

Beispiel



Varianten des Basis-Algorithmus

k-means [MacQueen 67]

Idee: die betroffenen Centroide werden direkt aktualisiert, wenn ein Punkt
seine Clusterzugehörigkeit ändert

- *K*-means hat im wesentlichen die Eigenschaften des Basis-Algorithmus
- *K*-means ist aber reihenfolgeabhängig

ISODATA

- basiert auf *k*-means
- Verbesserung des Ergebnisses durch Operationen wie
  - Elimination sehr kleiner Cluster
  - Verschmelzung und Aufspalten von Clustern
- Benutzer muss viele zusätzliche Parameter angeben

Diskussion

+ Effizienz
  Aufwand: O($k \cdot n$) für eine Iteration (k kann für kleine Anzahl von Clustern vernachlässigt werden), Anzahl der Iterationen ist im allgemeinen klein (~ 5 - 10).
+ einfache Implementierung

  $\Rightarrow$ *K*-means ist das populärste partitionierende Clustering-Verfahren

– Anfälligkeit gegenüber Rauschen und Ausreißern
  alle Objekte gehen ein in die Berechnung des Zentroids
– Cluster müssen konvexe Form haben
– die Anzahl k der Cluster ist oft schwer zu bestimmen
– starke Abhängigkeit von der initialen Zerlegung
  sowohl Ergebnis als auch Laufzeit

Auswahl repräsentativer Punkte

[Kaufman & Rousseeuw 1990]

- setze nur Distanzfunktion (*dist*) für Paare von Objekten voraus
- *Medoid*: ein zentrales Element des Clusters (repräsentativer Punkt)
- *Maß für die Kosten* (Kompaktheit) *eines Clusters C*

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

- *Maß für die Kosten* (Kompaktheit) *eines Clustering*

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

die Laufzeitkomplexität der erschöpfenden Suche ist $O(n^k)$

Überblick über k-medoid Algorithmen

PAM [Kaufman & Rousseeuw 1990]

- Greedy-Algorithmus:
  in jedem Schritt wird nur ein Medoid mit einem Nicht-Medoid vertauscht
- vertauscht in jedem Schritt das Paar (Medoid, Nicht-Medoid), das die größte Reduktion der Kosten TD bewirkt

CLARANS [Ng & Han 1994]

- zwei zusätzliche Parameter: *maxneighbor* und *numlocal*
- höchstens *maxneighbor* viele von zufällig ausgewählten Paaren (Medoid, Nicht-Medoid) werden betrachtet
- die erste Ersetzung, die überhaupt eine Reduzierung des *TD*-Wertes bewirkt, wird auch durchgeführt
- die Suche nach *k* „optimalen" Medoiden wird *numlocal* mal wiederholt

Algorithmus PAM

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

Algorithmus CLARANS

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


Vergleich von PAM und CLARANS

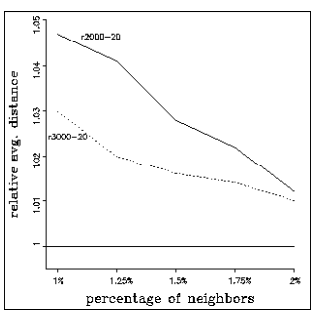
Laufzeitkomplexitäten

PAM: $O(n^3 + k(n-k)^2 * \#Iterationen)$

CLARANS: $O(numlocal * maxneighbor * \#Ersetzungen * n)$
praktisch $O(n^2)$

Experimentelle Untersuchung

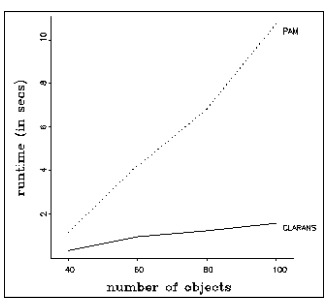
Qualität



TD(CLARANS)

TD(PAM)

Laufzeit



Erwartungsmaximierung (EM)

[Dempster, Laird & Rubin 1977]

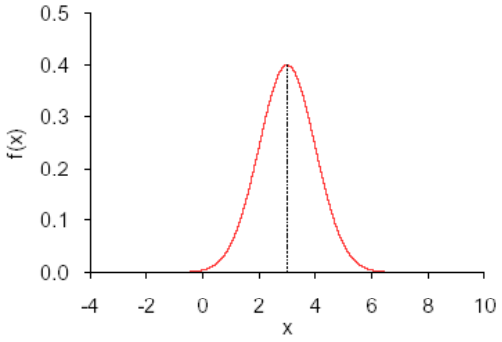
- Objekte sind Punkte $x = (x_1, ..., x_d)$ in einem euklidischen Vektorraum
- Ein Cluster wird durch eine Wahrscheinlichkeitsverteilung beschrieben
- typisch: Modell für einen Cluster ist eine multivariate Normalverteilung
- Repräsentation eines Clusters *C*
- Mittelwert μ_C aller Punkte des Clusters (Centroid)
- *d x d* Kovarianzmatrix Σ_C für die Punkte im Cluster *C*

- Wahrscheinlichkeitsdichte eines Clusters *C*

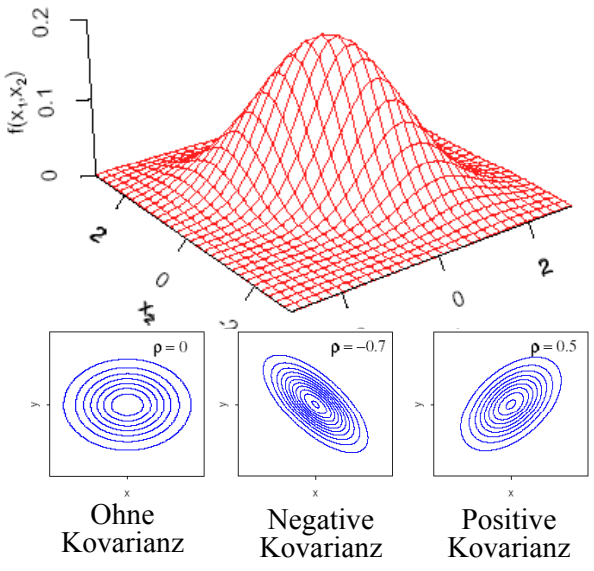
$$P(x \mid C) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_C \mid}} e^{-\frac{1}{2}\cdot(x-\mu_C)^T \cdot (\Sigma_C)^{-1} \cdot (x-\mu_C)}$$

Multivariate Normalverteilung

Univariate Normalverteilung



Bivariate Normalverteilung



5.2 Partitionierende Verfahren

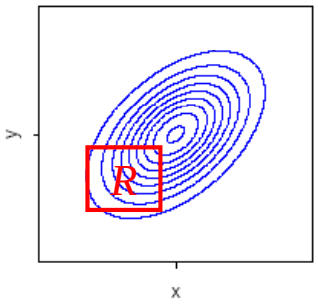
Idee des EM-Algorithmus:

- Jeder Punkt gehört zu mehreren (eigentlich *allen*) Clustern, jeweils mit unterschiedlicher Wahrscheinlichkeit, abh. v. P($x|C$)
- Algorithmus besteht wieder aus zwei alternierenden Schritten:
  - Zuordnung von Punkten zu Clustern (hier nicht absolut sondern relativ/nach Wahrscheinlichkeit)
  - Neuberechnung der Cluster-Repräsentanten (Gauß-Kurven)
- Alles muss auf eine stochastische Grundlage gestellt werden:
- Bei Berechnung der Clusterzentren (μ_C) muss berücksichtigt werden, dass Punkte Clustern nicht absolut sondern nur relativ zugeordnet sind
- Wie groß ist die Wahrscheinlichkeit der Clusterzugehörigkeit?

5.2 Partitionierende Verfahren

Jeder Cluster C_i wird durch eine Wahrscheinlichkeitsdichte Funktion (Normalverteilung) modelliert:

$$P(x \mid C_i) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_{C_i} \mid}} e^{-\frac{1}{2}\cdot(x-\mu_{C_i})^T\cdot(\Sigma_{C_i})^{-1}\cdot(x-\mu_{C_i})}$$



Dichtefunktion:

- Integral über den Gesamtraum ergibt 1
- Integral über Region *R* ergibt Wahrscheinlichkeit, dass in der Region ein beliebiger Punkt des Clusters liegt, bzw. den relativen Anteil (z.B. 30%) der Punkte des Clusters, die in *R* liegen

$$P(x|C_i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma C_i|}} e^{-\frac{1}{2}\cdot(x-\mu_{C_i})^T\cdot(\Sigma_{C_i})^{-1}\cdot(x-\mu_{C_i})}$$

Bedingte Wahrscheinlichkeit:

- Dies würde unter der Voraussetzung gelten, dass der Punkt *x* ausschließlich dem Cluster C_i zugeordnet wäre (was nicht stimmt)
- Deshalb Notation als *bedingte* Wahrscheinlichkeit

Bei *k* Gaussverteilungen (durch *k* Cluster) ergibt sich folgende Gesamt-Wahrscheinlichkeitsdichte:

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

wobei W_i der relative Anteil der Datenpunkte ist, der zum Cluster C_i gehört (z.B. 5%), was man auch als Gesamt-Wahrscheinlichkeit des Clusters $P(C_i)$ interpretieren kann.

Mit dem *Satz von Bayes* kann man die Wahrscheinlichkeit bestimmen, dass ein gegebener Punkt *x* zum Cluster C_i gehört, geschrieben als bedingte Wahrscheinlichkeit $P(C_i|x)$

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

Bedingte Wahrscheinlichkeiten

- Seien $A, B \subseteq \Omega$. Die *bedingte Wahrscheinlichkeit* von A unter B, $P(A|B)$, ist definiert als

$$P(A|B) = \begin{cases} 0 \ \ falls \ P(B) = 0 \\ \dfrac{P(A \cap B)}{P(B)} \ \ sonst \end{cases}$$

- A und B heißen *unabhängig*, wenn gilt $P(A|B) = P(A)$ und $P(B|A) = P(B)$.

Satz von Bayes

- Sei $A_1, \ldots, A_k$ eine disjunkte Zerlegung von Ω, so daß für mindestens ein i, $1 \le i \le k$, gilt: $P(A_i) > 0$ und $P(B|A_i) > 0$. Dann gilt für alle $1 \le j \le k$:

$$P(A_j|B) = \frac{P(B|A_j) \cdot P(A_j)}{P(B)}$$

- *a-priori*-Wahrscheinlichkeit: $P(A_i)$
- *a-posteriori*-Wahrscheinlichkeit: $P(A_i|B)$

- Maß für die Güte eines Clustering M

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

$\Rightarrow$ $E(M)$ soll maximiert werden.

- Anteil des Clusters an der Datenmenge:

$$W_i = P(C_i) = \frac{1}{n} \sum_{j=1}^{n} P(C_i \mid x_j)$$

- Mittelwert und Kovarianzmatrix der Gaußverteilung:

$$\mu_i = \left(\sum_{x \in D} x \cdot P(C_i|x)\right) / \left(\sum_{x \in D} P(C_i|x)\right)$$

$$\Sigma_i = \left(\sum_{x \in D} (x - \mu_i)(x - \mu_i)^{\mathrm{T}} \cdot P(C_i|x)\right) / \left(\sum_{x \in D} P(C_i|x)\right)$$

Algorithmus

```
ClusteringDurchErwartungsmaximierung(Punktmenge D, Integer k)
    Erzeuge ein „initiales“ Modell M’ = (C1’, ..., Ck’);
    repeat // „Neuzuordnung“
        Berechne P(x|Ci), P(x) und P(Ci|x) für jedes Objekt aus
          D und jede Gaußverteilung/jeden Cluster Ci;
        // „Neuberechnung des Modells“
        Berechne ein neues Modell M ={C1, ..., Ck} durch
          Neuberechnung von Wi, μC und ΣC für jedes i;
        M’ := M;
    until |E(M) - E(M’)| < ε;
    return M;
```


Diskussion

- Aufwand:

  O(*n* ∗ |M| * #*Iterationen*)

  Anzahl der benötigten Iterationen im allgemeinen sehr hoch
- Ergebnis und Laufzeit hängen (wie beim *k-means* und *k-medoid*) stark ab
  - von der initialen Zuordnung
  - von der „richtigen“ Wahl des Parameters *k*
- Modifikation für Partitionierung der Daten in *k disjunkte* Cluster:
  jedes Objekt nur demjenigen Cluster zuordnen, zu dem es am wahrscheinlichsten gehört.

Wahl des initialen Clusterings

Idee

- Clustering einer kleinen Stichprobe liefert im allgemeinen gute initiale Cluster einzelne Stichproben sind evtl. deutlich anders verteilt als die Grundgesamtheit

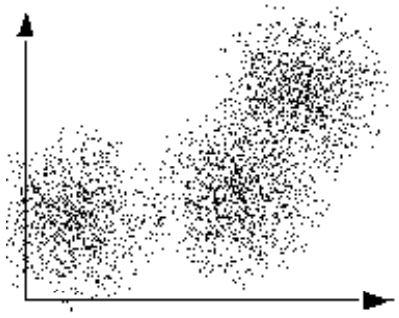
Methode [Fayyad, Reina & Bradley 1998]

- ziehe unabhängig voneinander m verschiedene Stichproben
- clustere jede der Stichproben

  $\Rightarrow$ m verschiedene Schätzungen für k Clusterzentren

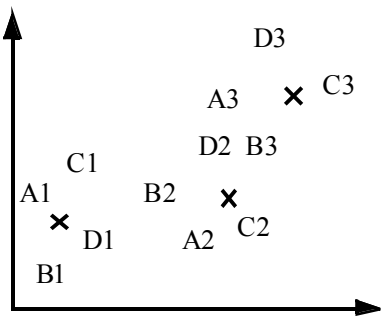
  $A = (A_1, A_2, \ldots, A_k)$, $B = (B_1, \ldots, B_k)$, $C = (C_1, \ldots, C_k)$, $\ldots$
- Clustere nun die Menge $DB = A \cup B \cup C..$
  mit m verschiedenen Stichproben A, B, C, $\ldots$ als Startkonfiguration
- Wähle von den m Clusterings dasjenige mit dem besten Wert bezüglich des zugehörigen Maßes für die Güte eines Clustering

Beispiel



Grundgesamtheit

$k = 3$ Gauß-Cluster



Clusterzentren

von $m = 4$ Stichproben

× *wahre Clusterzentren*

Wahl des Parameters k

Methode

- Bestimme für $k = 2, \ldots, n-1$ jeweils ein Clustering
- Wähle aus der Menge der Ergebnisse das „beste“ Clustering aus

Maß für die Güte eines Clusterings

- muss unabhängig von der Anzahl k sein
- bei k-means und k-medoid: TD^2 und TD sinken monoton mit steigendem k
- bei EM: E steigt monoton mit steigendem k
- Brauche ein von k unabhängiges Gütemaß für die k-means- und k-medoid-Verfahren

⇨ *Silhouetten-Koeffizient*

Silhouetten-Koeffizient [Kaufman & Rousseeuw 1990]

- sei $a(o)$ der Abstand eines Objekts o zum Repräsentanten seines Clusters und $b(o)$ der Abstand zum Repräsentanten des „zweitnächsten“ Clusters
- Silhouette $s(o)$ von o:

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

$-1 \leq s(o) \leq +1$

$s(o) \approx -1 / 0 / +1$: *schlecht / indifferent / gute Zuordnung*

- Silhouettenkoeffizient s_C eines Clustering durchschnittliche Silhouette aller Objekte
- Interpretation des Silhouettenkoeffizients

$s_C > 0{,}7$: starke Struktur,

$s_C > 0{,}5$: brauchbare Struktur, . . .

Grundlagen

Idee

- Cluster als Gebiete im *d*-dimensionalen Raum, in denen die Objekte dicht beieinander liegen
- getrennt durch Gebiete, in denen die Objekte weniger dicht liegen

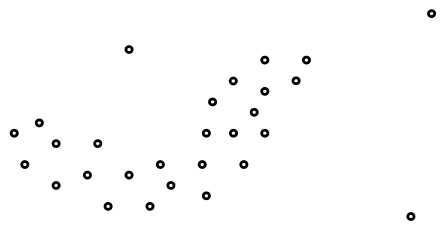
Anforderungen an dichtebasierte Cluster

- für jedes Objekt eines Clusters überschreitet die lokale Punktdichte einen gegebenen Grenzwert
- die Menge von Objekten, die den Cluster ausmacht, ist räumlich zusammenhängend

Intuition

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

ε $MinPts = 4$

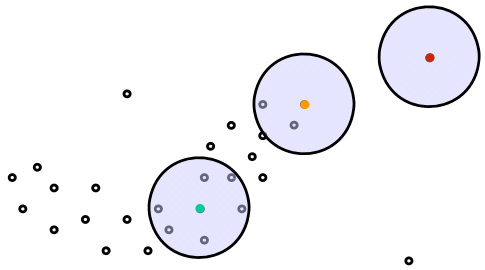


Intuition

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

ε $MinPts = 4$

- *Kernpunkt*

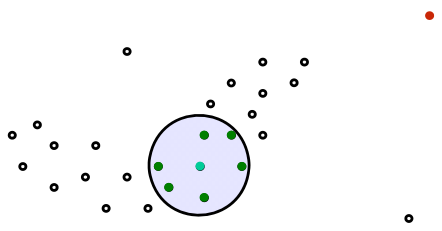


Intuition

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

ε $MinPts = 4$

- *Kernpunkt*
- *Direkt Dichte-Erreichbarkeit*

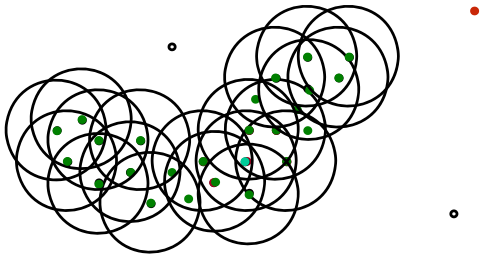


Intuition

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

ε $MinPts = 4$

- *Kernpunkt*
- *Direkt Dichte-Erreichbarkeit*
- *Dichte-Erreichbarkeit*



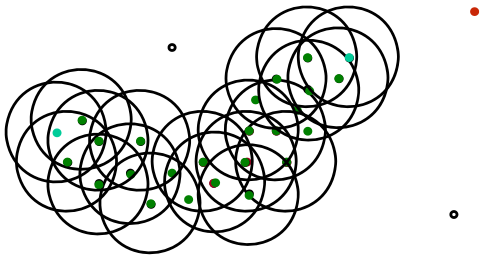
DATABASE SYSTEMS GROUP
5.3 Dichtebasiertes Clustering
LMU

Intuition

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

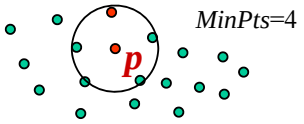
ε $MinPts = 4$

- *Kernpunkt*
- *Direkt Dichte-Erreichbarkeit*
- *Dichte-Erreichbarkeit*
- *Dichte-Verbundenheit*

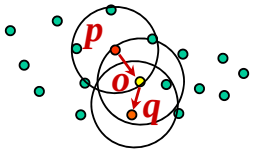


Formalisierung [Ester, Kriegel, Sander & Xu 1996]

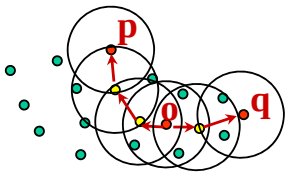
- Ein Objekt $p \in DB$ heißt *Kernobjekt*, wenn gilt:
  $|RQ(o,\varepsilon)| \geq MinPts$
  ($RQ(o,\varepsilon)$ siehe Kap. 2.2, Folie 37)



- Ein Objekt $p \in DB$ ist *direkt dichte-erreichbar* von $q \in DB$ bzgl. ε und *MinPts*, wenn gilt:
  $p \in RQ(q,\varepsilon)$ und q ist ein Kernobjekt in *DB*.



- Ein Objekt p ist *dichte-erreichbar* von q,
  wenn es eine Kette von direkt erreichbaren Objekten zwischen q und p gibt.
- Zwei Objekte p und q sind *dichte-verbunden*,
  wenn sie beide von einem dritten Objekt o aus dichte-erreichbar sind.



Formalisierung

Ein (*dichtebasierter*) *Cluster* C bzgl. ε und *MinPts* ist eine nicht-leere Teilmenge von *DB* für die die folgenden Bedingungen erfüllt sind:

Maximalität: $\forall p,q \in DB$: wenn $p \in C$ und q dichte-erreichbar von p ist, dann ist auch $q \in C$.

Verbundenheit: $\forall p,q \in C$: p ist dichte-verbunden mit q.

Formalisierung

Definition Clustering

Ein *dichtebasiertes Clustering* *CL* der Menge *DB* bzgl. ε und *MinPts* ist eine „vollständige" Menge von dichtebasierten Clustern bzgl. ε und *MinPts* in *DB*.

Definition Noise

Die Menge $Noise_{CL}$ („*Rauschen*") ist definiert als die Menge aller Objekte aus *DB*, die nicht zu einem der dichtebasierten Cluster $C \in CL$ gehören.

Grundlegende Eigenschaft

Sei *C* ein dichtebasierter Cluster und sei $p \in C$ ein Kernobjekt.
Dann gilt:

$$C = \{o \in DB \mid o \text{ dichte-erreichbar von } p \text{ bzgl. } \varepsilon \text{ und } MinPts\}.$$

Algorithmus DBSCAN

```
DBSCAN(Punktmenge DB, Real ε, Integer MinPts)
  // Zu Beginn sind alle Objekte unklassifiziert,
  // o.ClId = UNKLASSIFIZIERT für alle o ∈ DB

   ClusterId := nextId(NOISE);
   for i from 1 to |DB| do
      Objekt := DB.get(i);
      if Objekt.ClId = UNKLASSIFIZIERT then
         if ExpandiereCluster(DB, Objekt, ClusterId, ε, MinPts)
         then ClusterId:=nextId(ClusterId);
```

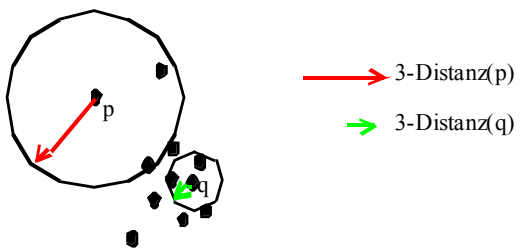
```
ExpandiereCluster(DB, StartObjekt, ClusterId, ε, MinPts): Boolean
seeds:= RQ(StartObjekt, ε);
if |seeds| < MinPts then // StartObjekt ist kein Kernobjekt
  StartObjekt.ClId := NOISE;
  return false;
// sonst: StartObjekt ist ein Kernobjekt
forall o ∈ seeds do o.ClId := ClusterId;
  entferne StartObjekt aus seeds;
  while seeds ≠ Empty do
    wähle ein Objekt o aus der Menge seeds;
    Nachbarschaft := RQ(o, ε);
    if |Nachbarschaft| ≥ MinPts then // o ist ein Kernobjekt
        for i from 1 to |Nachbarschaft| do
                p := Nachbarschaft.get(i);
                if p.ClId in {UNCLASSIFIED, NOISE} then
                        if p.ClId = UNCLASSIFIED then
                                füge p zur Menge seeds hinzu;
                        p.ClId := ClusterId;
    entferne o aus der Menge seeds;
return true;
```


Parameterbestimmung

Cluster: Dichte größer als die durch ε und *MinPts* spezifizierte „Grenzdichte“

Gesucht: der am wenigsten dichte Cluster in der Datenmenge

Heuristische Methode: betrachte die Distanzen zum *k*-nächsten Nachbarn.

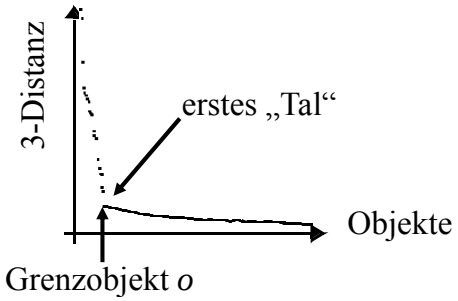


Funktion *k-Distanz*: Distanz eines Objekts zu seinem *k*-nächsten Nachbarn

k-Distanz-Diagramm: die *k*-Distanzen aller Objekte absteigend sortiert

Parameterbestimmung

Beispiel eines *k*-Distanz-Diagramms

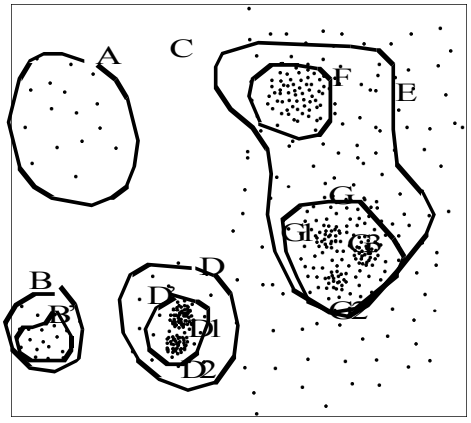


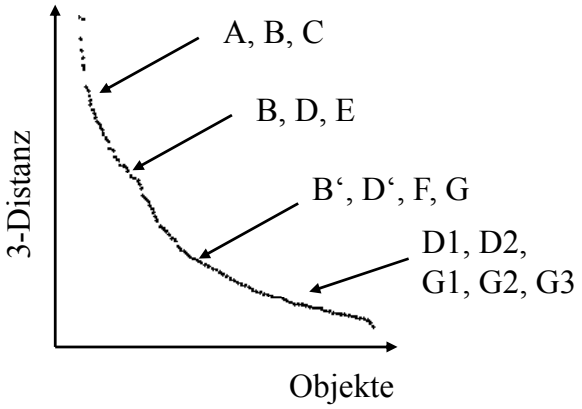
Heuristische Methode

- Benutzer gibt einen Wert für *k* vor (Default ist *k* = 2**d* - *1*), *MinPts* := *k*+1.
- System berechnet das *k*-Distanz-Diagramm und zeigt das Diagramm an.
- Der Benutzer wählt ein Objekt *o* im *k*-Distanz-Diagramm als Grenzobjekt aus, ε := *k*-Distanz(*o*).

Probleme der Parameterbestimmung

- hierarchische Cluster
- stark unterschiedliche Dichte in verschiedenen Bereichen des Raumes
- Cluster und Rauschen sind nicht gut getrennt

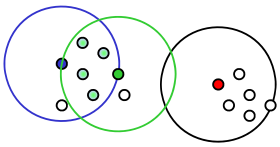




Shared Nearest Neighbor (SNN) Clustering

- DBSCAN
  - Erkennt Cluster unterschiedlicher Form und Größe
  - Hat Probleme bei Clustern mit unterschiedlicher Dichte
- Verbesserung: anderer Ähnlichkeitsbegriff
  - Ähnlichkeit zwischen zwei Objekten, wenn sie beide sehr nahe zu einer Referenzmenge *R* sind
  - Ähnlichkeit wird durch die Referenzmenge *R* "bestätigt"
  - Ähnlichkeit z.B. durch die Anzahl gemeinsamer nächster Nachbarn definieren (d.h. *R* ist die Menge der nächsten Nachbarn)
  - Shared Nearest Neighbor (SNN) Ähnlichkeit:
    - SNN_k-similarity(p,q) = $|NN(p, k) \cap NN(q, k)|$
    - $NN(o, k)$ = Menge der k-nächsten Nachbarn von o (vgl. Kap. 2.2)

$k = 6$

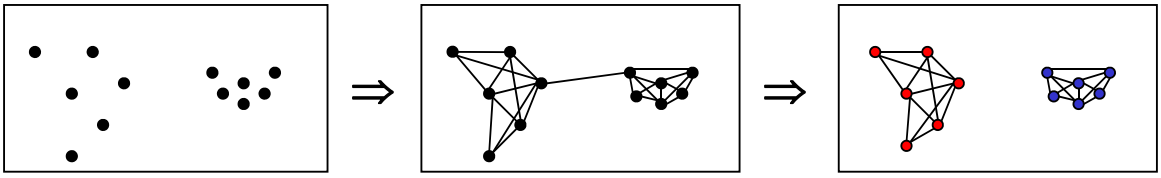


SNN_6-similarity(● ,●) = 4

SNN_6-similarity(● ,●) = 0

Einfaches SNN-Clustering [Jarvis, Patrick 73]:

1. Berechnung der Ähnlichkeitsmatrix und des Ähnlichkeitsgraphen
   - für alle Objekt-Paare $p,q \in DB$: berechne SNN_k-similarity(p,q)
   - SNN_k-Ähnlichkeitsgraph:
     - Knoten = Objekte
     - Kante zwischen jedem Objektpaar p,q mit Gewicht SNN_k-similarity(p,q)
   - Keine Kanten mit Gewicht 0
2. Generiere Cluster
   - Lösche alle Kanten, deren Gewicht unterhalb eines Grenzwerts τ liegen
   - Cluster = verbundene Komponenten im resultierenden Graphen

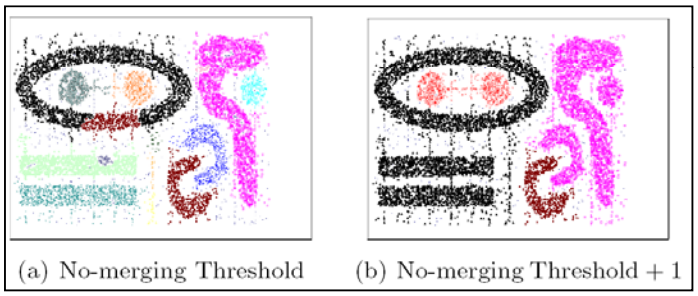


Datensatz | SNN_5 Ähnlichkeitsgraph | Generiere Cluster

5.3 Dichtebasiertes Clustering

Problem:

- Threshold τ schwer zu bestimmen
- kleine Variationen führen zu stark unterschiedlichen Ergebnissen



(a) No-merging Threshold (b) No-merging Threshold + 1

Bild aus:
[Ertöz, Steinbach, Kumar 03]

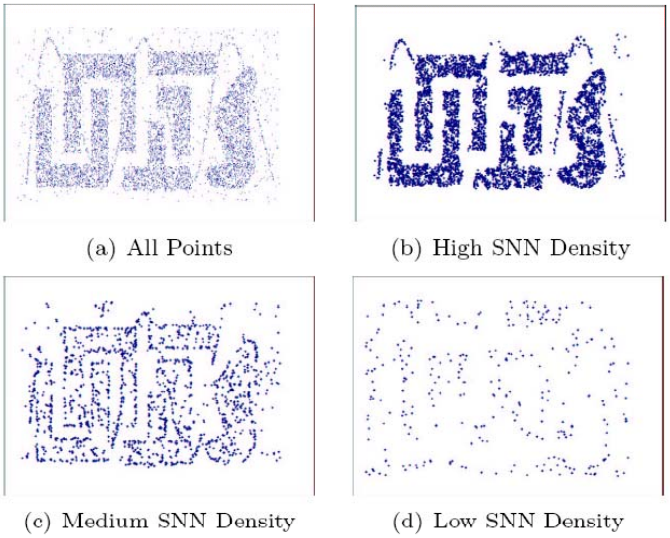
Lösung [Ertöz, Steinbach, Kumar 03]

- Kombiniere SNN-Ähnlichkeit mit dichtebasierten Konzepten
- SNN-Dichte:
  Anzahl der Punkte innerhalb eines spezifizierten Radius ε bzgl. SNN-Ähnlichkeit
  $SNN_k\text{-density}(p,\varepsilon) = |\{q \mid SNN_k\text{-similarity}(p,q) \geq \varepsilon\}|$

5.3 Dichtebasiertes Clustering

SNN-Dichte

- Beispiel:
  - 10 000 Daten (Bild (a))
    $k = 50$, $\varepsilon = 20$
  - Bild (b): "Kernpunkte"
    alle Punkte mit SNN-Dichte ≥ 34
  - Bild (c): "Randpunkte"
    alle Punkte mit SNN-Dichte ≥ 17
  - Bild (d): "Rauschen"
    alle Punkte mit SNN-Dichte < 17



(a) All Points (b) High SNN Density

(c) Medium SNN Density (d) Low SNN Density

Bild aus: [Ertöz, Steinbach, Kumar 03]

- Analogie zu DBSCAN: $\varepsilon = 20$, *minPts* = 34
- Kernpunkt *p*: mehr als *minPts* Punkte haben 20 oder mehr der 50 nächsten Nachbarn mit *p* gemeinsam

SNN-Clustering Algorithmus [Ertöz, Steinbach, Kumar 03]

Eingabe: *k*, ε, *minPts*

1. Berechne Ähnlichkeitsmatrix und –graph
   (siehe einfaches SNN-Clustering)
2. Berechne die SNN_k-Dichte für jeden Punkt bzgl. ε
3. Bestimme Kernpunkte bzgl. *minPts*
   (alle Punkte mit einer SNN-Dichte ≥ *minPts*)
4. Vereinige Kernpunkte *p*,*q*, wenn SNN_k-similarity(*p*,*q*) ≥ ε
5. Ordne Nicht-Kernpunkt *p* einem Cluster zu, wenn es einen Kernpunkt *q* gibt, mit SNN_k-similarity(*q*,*p*) ≥ ε
6. Alle anderen Nicht-Kernpunkte sind Rauschen

→ DBSCAN mit SNN-Ähnlichkeit

Diskussion

- Unterschied zu DBSCAN
  - DBSCAN mit Euklidischer Distanz: nur Cluster, die dichter sind als der Grenzwert (spezifiziert durch *minPts* und ε)
  - SNN-Dichte eines Punktes *p*: Anzahl der Punkte, die mind. ε nächste Nachbarn mit *p* gemeinsam haben
  - ⇒ unabhängig von der eigentlichen Dichte
- Parametrisierung
  - Wahl von *k* ist kritisch:
    - Zu klein: auch relativ gleichverteilte Cluster werden wegen lokalen Variationen gesplittet ⇒ viele kleine Cluster
    - Zu groß: wenige große, gut-separierte Cluster
  - *minPts*, ε < *k*

Grundlagen

Ziel

Konstruktion einer Hierarchie von Clustern (*Dendrogramm*), so dass immer die Cluster mit minimaler Distanz verschmolzen werden

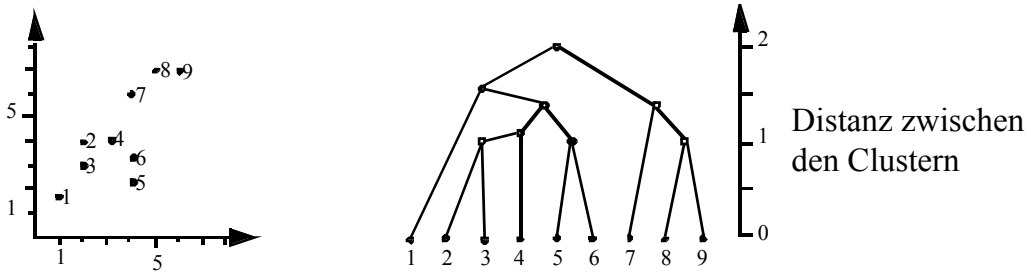
Dendrogramm

ein Baum, dessen Knoten jeweils einen Cluster repräsentieren, mit folgenden Eigenschaften:

- die Wurzel repräsentiert die ganze DB
- die Blätter repräsentieren einzelne Objekte
- ein innerer Knoten repräsentiert einen Cluster bestehend aus allen Objekten des darunter liegenden Teilbaums

Grundlagen

Beispiel eines Dendrogramms



Typen von hierarchischen Verfahren

Bottom-Up Konstruktion des Dendrogramms (*agglomerative*)

Top-Down Konstruktion des Dendrogramms (*divisive*)

Algorithmus Single-Link [Jain & Dubes 1988]

Agglomeratives hierarchisches Clustering

1. Bilde initiale Cluster, die jeweils aus einem Objekt bestehen, und bestimme die Distanzen zwischen allen Paaren dieser Cluster.
2. Bilde einen neuen Cluster aus den zwei Clustern, welche die geringste Distanz zueinander haben.
3. Bestimme die Distanz zwischen dem neuen Cluster und allen anderen Clustern.
4. Wenn sich alle Objekte in einem einzigen Cluster befinden: Fertig, andernfalls wiederhole ab Schritt 2.

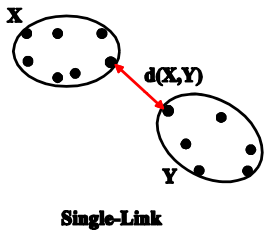
Distanzfunktionen für Cluster

- Die Verfahren unterscheiden sich anhand ihrer Distanzfunktionen für Cluster
  - Single Link
  - Complete Link
  - Average Link
- Sei eine Distanzfunktion *dist*(*x*,*y*) für Paare von Objekten gegeben
- Seien *X*, *Y* Cluster, d.h. Mengen von Objekten.

Single-Link Distanz

Definition:

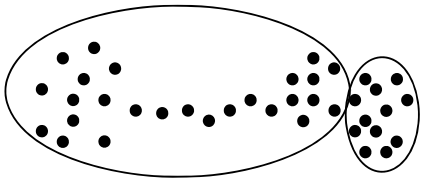
$$distSL(X,Y) = \min_{x \in X, y \in Y} dist(x,y)$$



Single-Link

Eigenschaften:

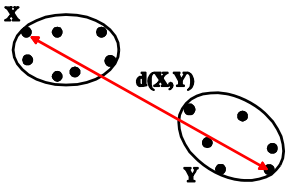
- Effiziente Implementierung (z.B. SLINK): O(n²)
- Single-Link Effekt: „kettenförmige" Cluster, Cluster werden durch wenige, kettenförmig verteilte Objekte vereinigt
  - Cluster mit starker Streuung
  - Cluster mit langgezogener Struktur



Complete-Link Distanz

Definition:

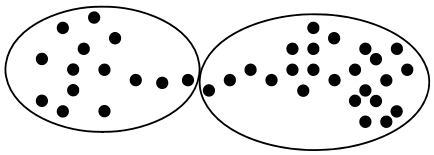
$$distCL(X,Y) = \max_{x \in X, y \in Y} dist(x,y)$$



Complete-Link

Eigenschaften:

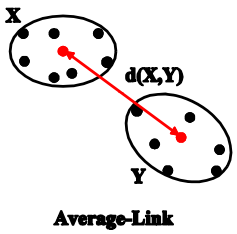
- Effiziente Implementierung (z.B. CLINK): O(n²)
- Complete-Link Effekt
  - Kleine, stark abgegrenzte Cluster
  - Gleichgroße, konvexe Cluster



Average-Link Distanz

Definition:

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



Eigenschaften:

- Keine effiziente Implementierung
- Kompromiss zwischen Single- und Complete-Link Ansatz

Diskussion

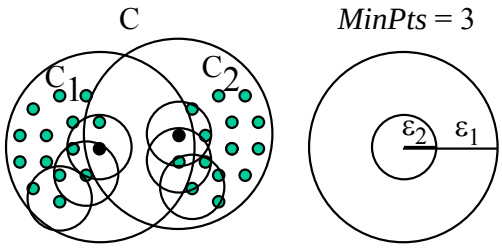
+ erfordert keine Kenntnis der Anzahl *k* der Cluster
+ findet nicht nur ein flaches Clustering, sondern eine ganze Hierarchie
+ ein einzelnes Clustering kann aus dem Dendrogramm gewonnen werden, z.B. mit Hilfe eines horizontalen Schnitts durch das Dendrogramm (erfordert aber wieder Anwendungswissen)

- Entscheidungen können nicht zurückgenommen werden
- Single-Link-Effekte, Complete-Link-Effekte
- Ineffizienz: Laufzeitkomplexität von mindestens $O(n^2)$ für n Objekte

Dichtebasiertes hierarchisches Clustering

[Ankerst, Breunig, Kriegel & Sander 1999]

- für einen konstanten *MinPts*-Wert sind dichte-basierte Cluster bzgl. eines kleineren ε vollständig in Clustern bzgl. eines größeren ε enthalten



- in einem DBSCAN-ähnlichen Durchlauf gleichzeitig das Clustering für verschiedene Dichte-Parameter bestimmen
- zuerst die dichteren Teil-Cluster, dann den dünneren Rest-Cluster
- kein Dendrogramm, sondern eine auch noch bei sehr großen Datenmengen übersichtliche Darstellung der Cluster-Hierarchie

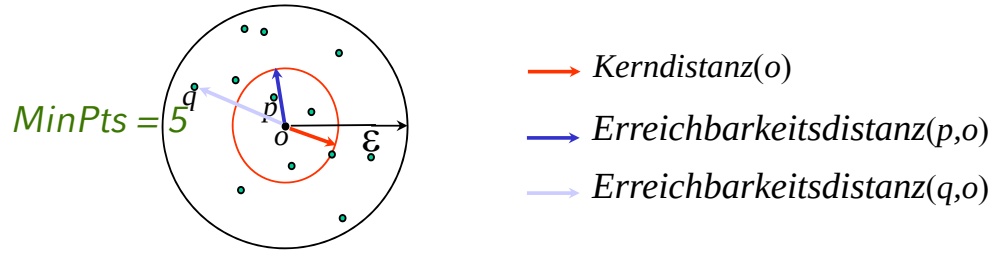
Grundbegriffe

Kerndistanz eines Objekts *o* bzgl. ε und *MinPts*

$$Kerndistanz_{\varepsilon,MinPts}(o) = \begin{cases} UNDEFINIERT, & wenn\ |RQ(o,\varepsilon)| < MinPts \\ MinPtsDistanz(o), & sonst \end{cases}$$

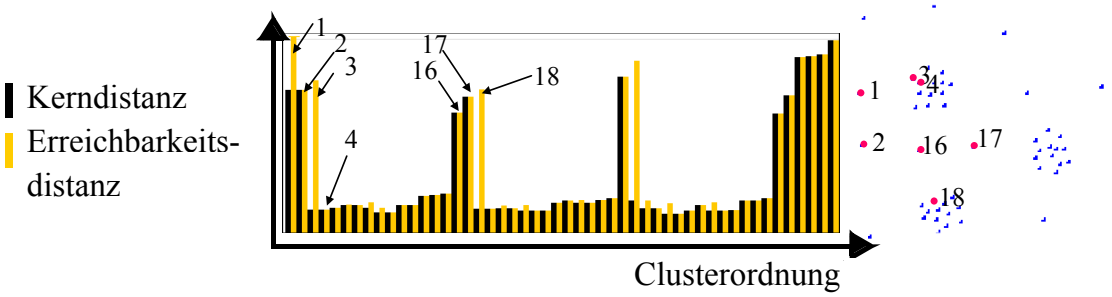
Erreichbarkeitsdistanz eines Objekts *p* relativ zu einem Objekt *o*

$$Erreichbarkeitsdistanz_{\varepsilon,MinPts}(p,o) = \begin{cases} UNDEFINIERT & wenn\ |RQ(o,\varepsilon)| < MinPts \\ \max\{Kerndistanz(o), dist(o,p)\}, & sonst \end{cases}$$



Clusterordnung

- OPTICS liefert nicht direkt ein (hierarchisches) Clustering, sondern eine „Clusterordnung“ bzgl. e und *MinPts*
- *Clusterordnung* bzgl. e und *MinPts*
  - beginnt mit einem beliebigen Objekt
  - als nächstes wird das Objekt besucht, das zur Menge der bisher besuchten Objekte die minimale Erreichbarkeitsdistanz besitzt



Algorithmus OPTICS

Datenstrukturen

- SeedList
  - speichert Punkte mit „aktueller“ Erreichbarkeitsdistanz aufsteigend sortiert
- ClusterOrder
  - resultierende Clusterordnung wird schrittweise aufgebaut

Hauptschleife:

```
SeedList = ∅;
WHILE es gibt noch unmarkierte Objekte in DB DO
   IF SeedList = ∅
   THEN
     füge beliebiges noch unmarkiertes Objekt in ClusterOrder ein mit
     Erreichbarkeitsdistanz ∞;
   ELSE
     füge erstes Objekt aus der SeedList mit aktueller Erreichbarkeitsdistanz in
     ClusterOrder ein;
   // sei obj das zuletzt in ClusterOrder eingefügte Objekt
   markiere obj als bearbeitet;
   FOR ALL neighbor ∈ RQ(obj, ε) DO
     // insert/update neighbor in SeedList mit referenzobjekt obj
     SeedList.update(neighbor, obj);
```

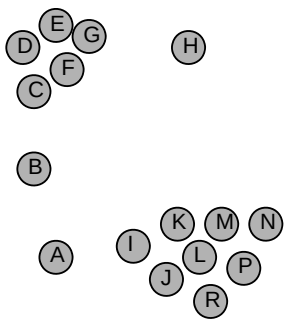
Algorithmus OPTICS

Einfügen/Updaten eines Objekts *o* in SeedList

- Beachte: Für alle Objekte *p* in SeedList ist die „aktuelle" Erreichbarkeitsdistanz *p*.rdist gespeichert.
- SeedList ist nach *p*.rdist aufsteigend sortiert (als Heap organisiert)
- Referenzobjekt: *obj*

```
SeedList :: update(o, obj)
Berechne Erreichbarkeitsdistanz_{ε,MinPts}(o, obj) =: rdistneu_o;
IF o ist bereits in SeedList THEN
   IF rdistneu_o ≤ o.rdist THEN
     o.rdist := rdistneu_o;
     verschiebe o in SeedList (nach vorne); // aufsteigen im Heap
ELSE
   // o ist noch nicht in SeedList => normales Einfügen in Heap
   füge o mit o.rdist := rdistneu_o in SeedList ein;
```

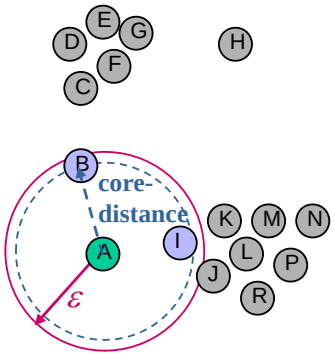

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3



seed list:

5.4 Hierarchische Verfahren

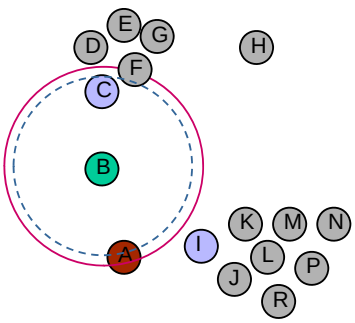
- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

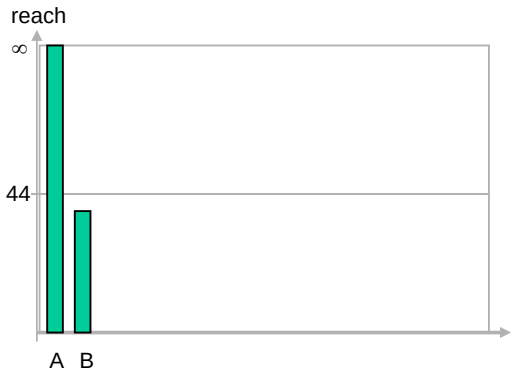


seed list: (B,40) (I, 40)

5.4 Hierarchische Verfahren

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

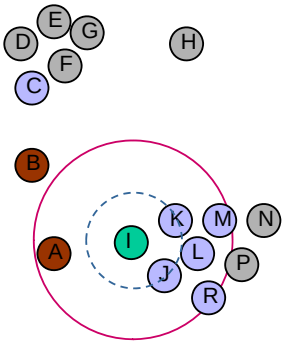




seed list: (I, 40) (C, 40)

5.4 Hierarchische Verfahren

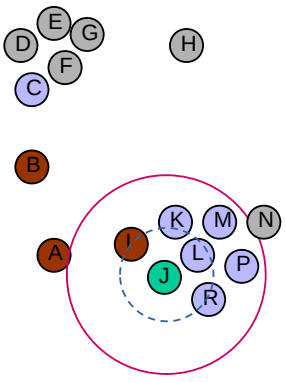
- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3



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

5.4 Hierarchische Verfahren

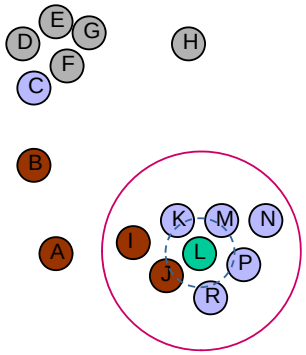
- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3



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

5.4 Hierarchische Verfahren

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3



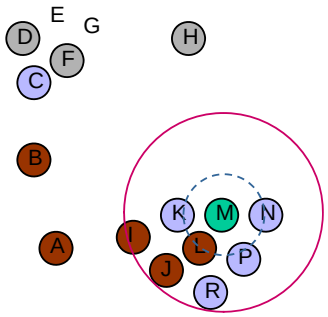
seed list: (M, 18) (K, 18) (R, 20) (P, 21) (N, 35) (C, 40)

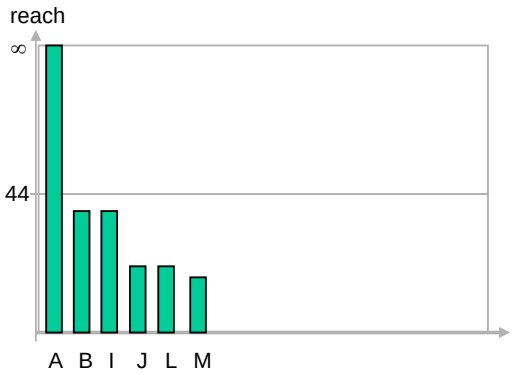
DATABASE SYSTEMS GROUP

5.4 Hierarchische Verfahren

LMU

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

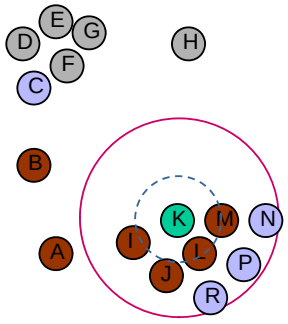


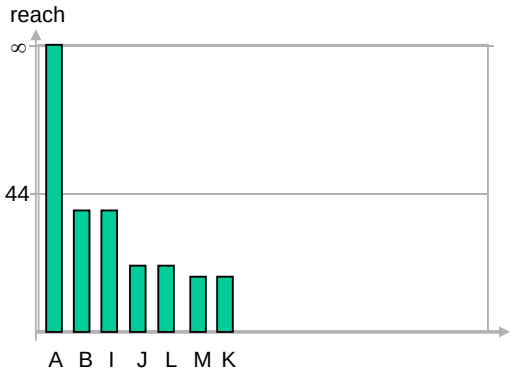


seed list: (K, 18) (N, 19) (R, 20) (P, 21) (C, 40)

5.4 Hierarchische Verfahren

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3





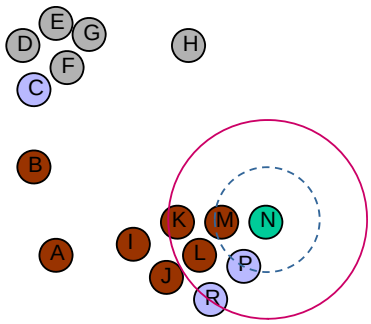
seed list: (N, 19) (R, 20) (P, 21) (C, 40)

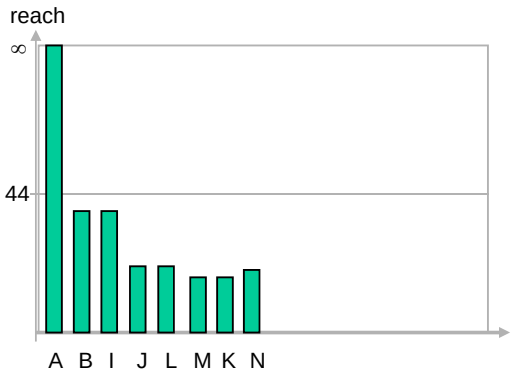
DATABASE SYSTEMS GROUP

5.4 Hierarchische Verfahren

LMU

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

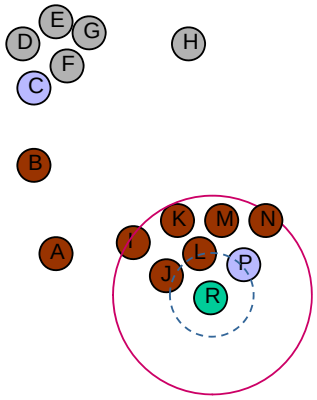


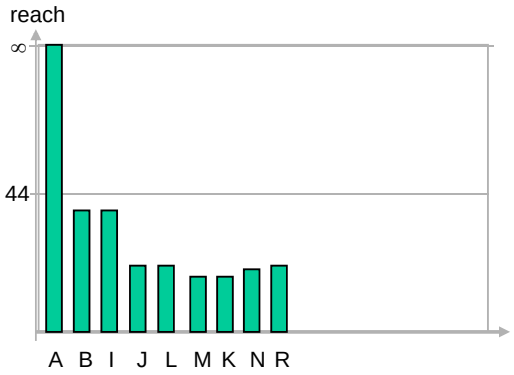


seed list: (R, 20) (P, 21) (C, 40)

5.4 Hierarchische Verfahren

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

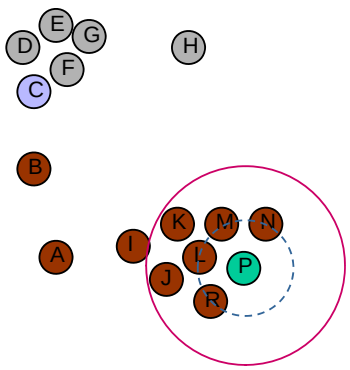


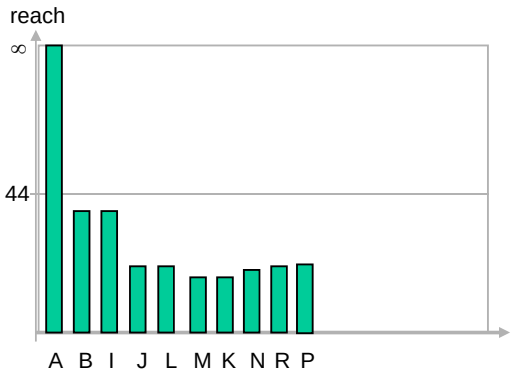


seed list: (P, 21) (C, 40)

5.4 Hierarchische Verfahren

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

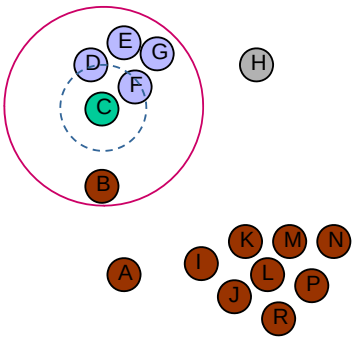




seed list: (C, 40)

5.4 Hierarchische Verfahren

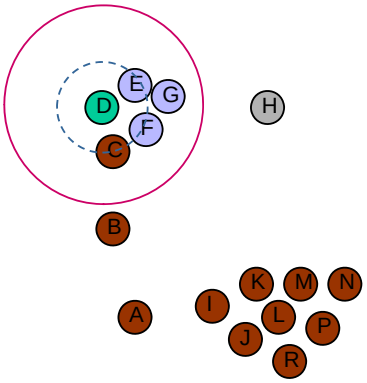
- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3



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

5.4 Hierarchische Verfahren

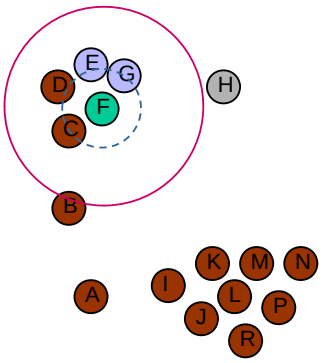
- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

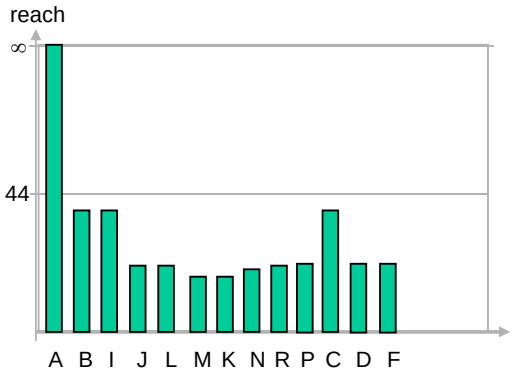


seed list: (F, 22) (E, 22) (G, 32)

5.4 Hierarchische Verfahren

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

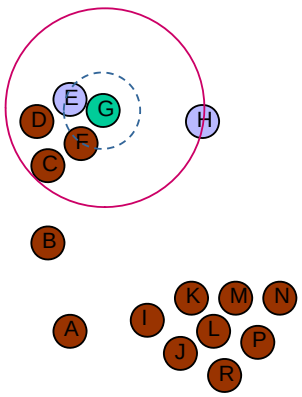


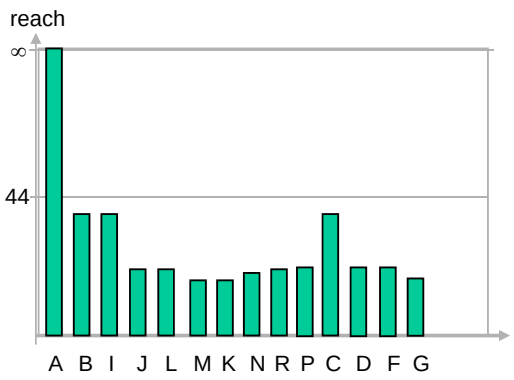


seed list: (G, 17) (E, 22)

5.4 Hierarchische Verfahren

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

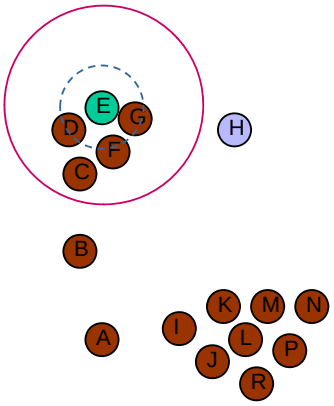


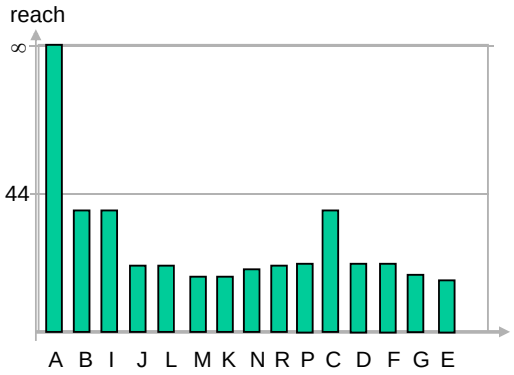


seed list: (E, 15) (H, 43)

5.4 Hierarchische Verfahren

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

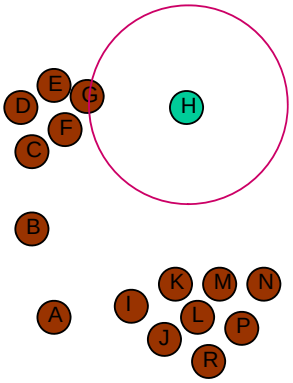


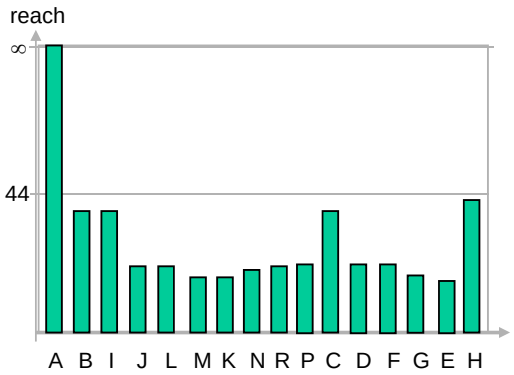


seed list: (H, 43)

5.4 Hierarchische Verfahren

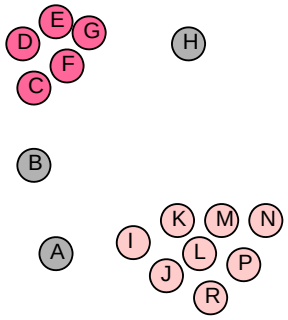
- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

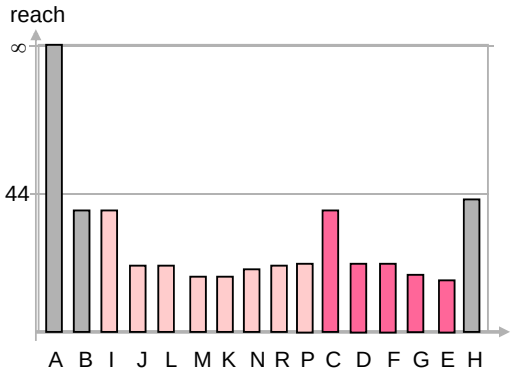




seed list: -

- Example Database (2-dimensional, 16 points)
- ε= 44, *MinPts* = 3

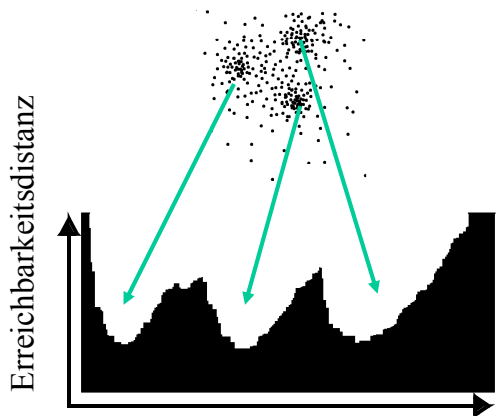


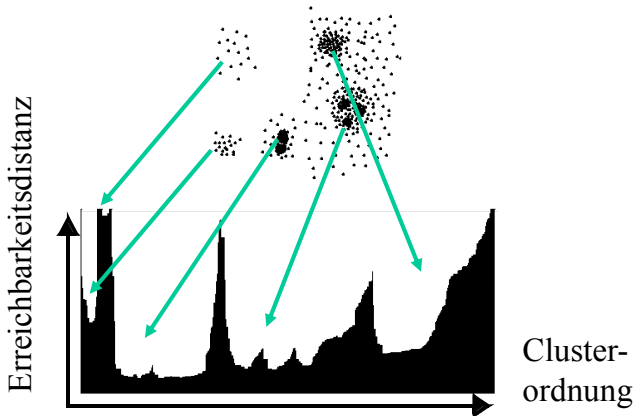


Erreichbarkeits-Diagramm

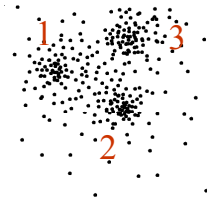
Zeigt die Erreichbarkeitsdistanzen (bzgl. ε und *MinPts*) der Objekte als senkrechte, nebeneinanderliegende Balken

in der durch die Clusterordnung der Objekte gegebenen Reihenfolge





Parameter-Sensitivität



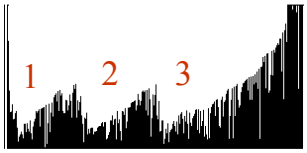
MinPts = 10, ε = 10

optimale Parameter

MinPts = 10, ε = 5

kleineres ε

MinPts = 2, ε = 10



kleineres *MinPts*

Clusterordnung ist robust gegenüber den Parameterwerten
gute Resultate wenn Parameterwerte „groß genug“

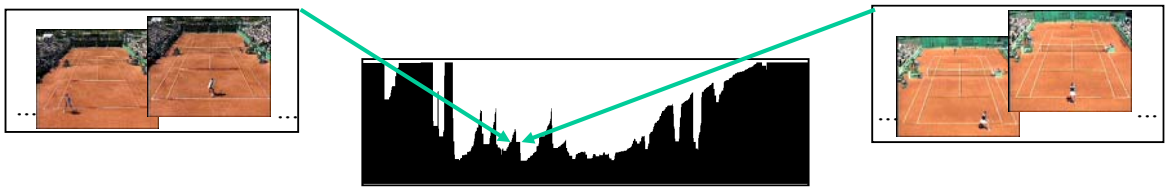
Heuristische Parameter-Bestimmung

ε

- wähle größte *MinPts*-Distanz aus einem Sample oder
- berechne durchschnittliche *MinPts*-Distanz für gleichverteilte Daten

MinPts

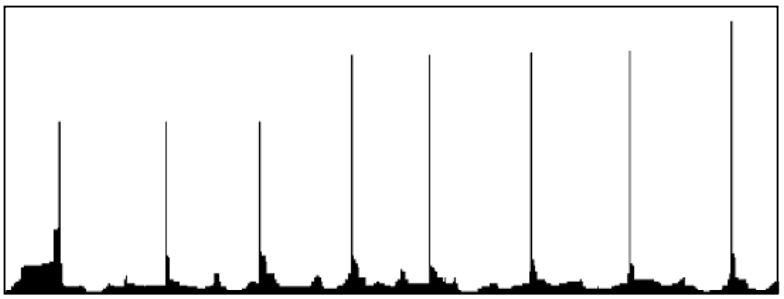
- glätte Erreichbarkeits-Diagramm
- vermeide “single-” bzw. “*MinPts*-link” Effekt



Manuelle Analyse der Cluster

Mit Erreichbarkeits-Diagramm

- gibt es Cluster?
- wieviele Cluster?
- sind die Cluster hierarchisch geschachtelt?
- wie groß sind die Cluster?

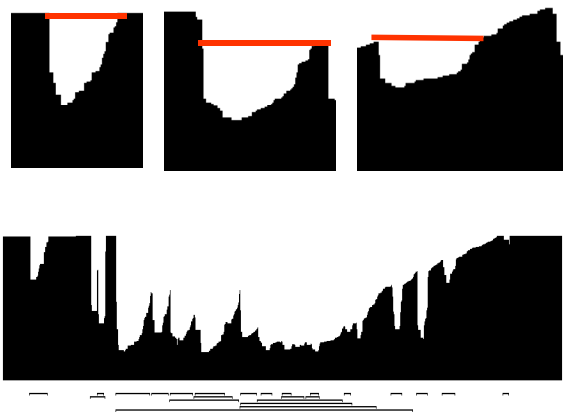


Erreichbarkeits-Diagramm

Automatisches Entdecken von Clustern

ξ-Cluster [Ankerst, Breunig, Kriegel, Sander 99]

- Teilsequenz der Clusterordnung
- beginnt in einem Gebiet ξ-steil *abfallender* Erreichbarkeitsdistanzen
- endet in einem Gebiet ξ-steil *steigender* Erreichbarkeitsdistanzen bei etwa demselben absoluten Wert
- enthält mindestens *MinPts* Punkte



Automatisches Entdecken von Clustern

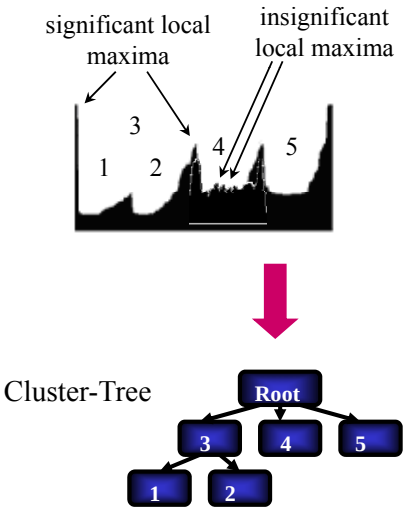
ClusterTree [Sander, Qin, Lu, Niu, Kovarsky 02]

Cluster sind geteilt durch "signifikante" lokale Maxima:

- *Minimale Anzahl der Punkte zwischen 2 Maxima*

  Richtwert: 0.5 % der Datenbank
- *Verhältnis zwischen Erreichbarkeitsdistanz des lokalen Maximas und der durchschnittlichen Erreichbarkeitsdistanzen links und rechts des Maximas in der Clusterordnung*

  Richtwert: 0.75



Automatisches Entdecken von Clustern

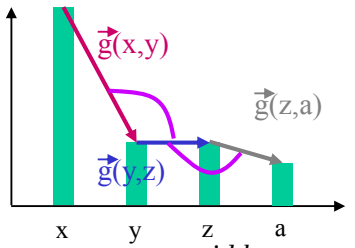
GradientClustering

[Brecheisen, Kriegel, Kröger, Pfeifle 04]

- Teilsequenzen der Clusterordnung
- beginnt/endet mit "Inflexion Point"
  - Gradientvektor
  - Inflexion Index
  - Inflexion Point, wenn II(o) > t
  - Gradient Determinante
  - Fallunterscheidung:

    $II(o) > t$ and $GD(o) > 0$
    $\Rightarrow$ Startpunkt oder erster Punkt außerhalb des Clusters

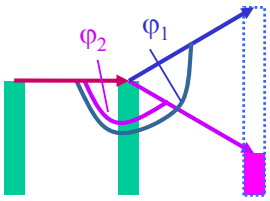
    $II(o) > t$ and $GD(o) < 0$
    $\Rightarrow$ Endpunkt oder zweiter Punkt innerhalb des Clusters



$$\vec{g}(x,y) = \begin{pmatrix} width \\ y.R - x.R \end{pmatrix}$$

$$II(y) = \cos \varphi(\vec{g}(x,y), \vec{g}(y,z))$$

$$GD(\vec{g}(x,y), \vec{g}(y,z)) = \begin{vmatrix} width & -width \\ y.R - x.R & y.R - z.R \end{vmatrix}$$



$$\cos \varphi_1 = \cos \varphi_2$$

5.5 Besondere Anwendungen

Übersicht

5.5.1 Clustering kategorischer Attribute

Grundlagen, Algorithmus *k*-modes

5.5.2 Verallgemeinertes dichtebasiertes Clustering

Grundlagen, Algorithmus GDBSCAN, Beispiele

5.5.1 Clustering mit kategorischen Attributen

Grundlagen [Huang 1997]

- *k*-medoid-Algorithmus wesentlich langsamer als *k*-means- Algorithmus
- *k*-means-Verfahren nicht direkt für kategorische Attribute anwendbar

gesucht ist ein Analogon zum Centroid eines Clusters

- Numerische Attribute
  Centroid $\bar{x}$ einer Menge *C* von Objekten minimiert $TD(C,\bar{x}) = \sum_{p \in C} dist(p,\bar{x})$
- Kategorische Attribute
- Mode *m* einer Menge *C* von Objekten minimiert $TD(C,m) = \sum_{p \in C} dist(p,m)$
  (*m* ist nicht unbedingt ein Element der Menge *C*)
- $m = (m_1, ..., m_d)$, *dist* eine Distanzfunktion für kategorische Attribute, z.B.

$$dist(x,y) = \sum_{i=1}^{d} \delta(x_i, y_i) \text{ mit } \delta(x_i, y_i) = \begin{cases} 0, & falls\ x_i = y_i \\ 1, & sonst \end{cases}$$

Bestimmung des Modes

- Die Funktion $TD(C,m) = \sum_{p \in C} dist(p,m)$ wird genau dann minimiert, wenn für $m = (m_1, ..., m_d)$ und für alle Attribute A_i, $i = 1,..., d$, gilt:

  Es gibt in A_i keinen häufigeren Attributwert als m_i

- Der Mode einer Menge von Objekten ist nicht eindeutig bestimmt.

- Beispiel

  Objektmenge {(a, b), (a,c), (c, b), (b,c)}
  (a, b) ist ein Mode
  (a, c) ist ein Mode

Algorithmus k-modes

- Initialisierung

  keine zufällige Partitionierung
  sondern *k* Objekte aus der Datenmenge als initiale *Modes*

- Cluster-Repräsentanten

  Mode anstelle des Centroids

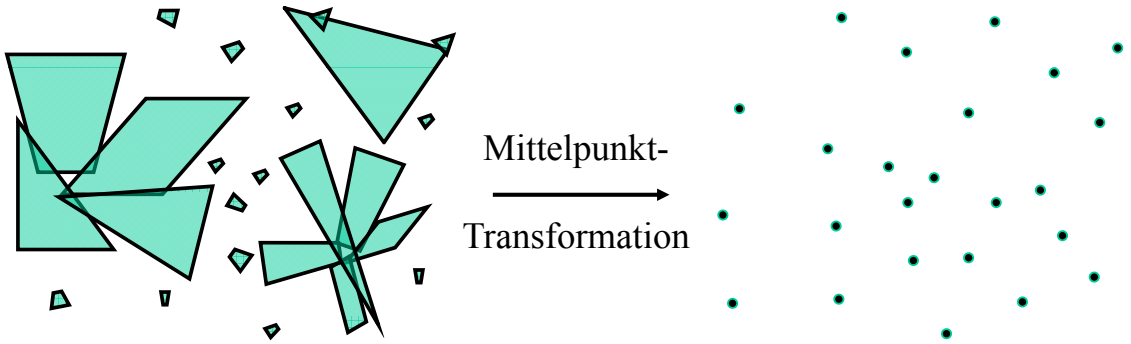
- Distanzfunktion

  anstelle der quadrierten euklidischen Distanz

  Distanzfunktion für Datensätze mit kategorischen Attributen

5.5.2 Verallgemeinertes dichtebasiertes Clustering

Clustering ausgedehnter Objekte



Attribute

Berücksichtigung der Fläche und nicht-räumlicher

natürlicher Begriff der Verbundenheit

DATABASE SYSTEMS GROUP

5.5.2 Verallgemeinertes dichtebasiertes Clustering

LMU

Algorithmus GDBSCAN [Sander, Ester, Kriegel & Xu 1998]

Grundidee für dichte-basierte Cluster :

"*ε-Nachbarschaft* enthält mindestens *MinPts* Punkte"

"Distanz ≤ ε"

$NPred(o,p)$
reflexiv, symmetrisch
für Paare von Objekten

Verallgemeinerte Nachbarschaft
$N_{NPred}(o) = \{p \mid NPred(o, p)\}$

"$|N_\varepsilon| \geq MinPts$"

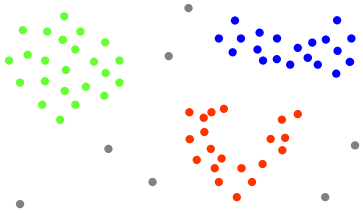
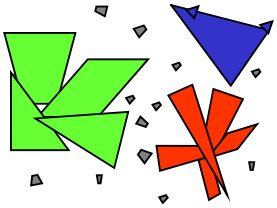
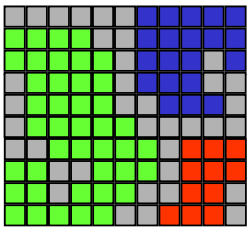
$MinWeight(N)$
beliebiges Prädikat für
Mengen von Objekten

Verallgemeinerte minimale Kardinalität
$MinWeight(N_{NPred}(o))$

"*NPred-Nachbarschaft* hat mindestens das "Gewicht" *MinWeight*"

5.5.2 Verallgemeinertes dichtebasiertes Clustering

Beispiele

			
NPred	dist(p,q) ≤ ε	intersect(p,q)	Nachbarzelle und ähnliche Farbe
MinWeight	cardinality(. . .) ≥ MinPoints	Summe der Flächen ≥ 5 % der Gesamtfläche	true

5.5.2 Verallgemeinertes dichtebasiertes Clustering

Algorithmus GDBSCAN

- dasselbe algorithmische Schema wie DBSCAN
- anstelle einer RQ(o,ε)-Anfrage eine N_{NPred}-Anfrage
- anstelle der Bedingung |RQ(o,ε)| ≥ *MinPts*
- das *MinWeight*-Prädikat auswerten
- Laufzeitkomplexität O(n log n) bei geeigneter Unterstützung der N_{NPred}-Anfrage
- Beliebige Nachbarschaftsprädikate denkbar