3. Clustering

Inhalt dieses Kapitels

3. Clustering

Inhalt dieses Kapitels

3.5 Datenbanktechniken zur Leistungssteigerung

Ziel

Bisher

- kleine Datenmengen
- hauptspeicherresident

Jetzt

- sehr große Datenmengen, die nicht in den Hauptspeicher passen
- Daten auf Sekundärspeicher
  Zugriffe viel teurer als im Hauptspeicher
- effiziente Algorithmen erforderlich
  d.h. Laufzeitaufwand höchstens O(n log n)

Skalierbarkeit von Clustering-Algorithmen

3.5 Datenbanktechniken zur Leistungssteigerung

Idee

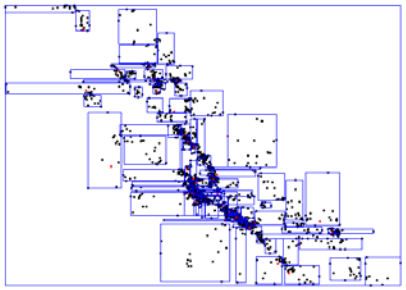
Verwendung von räumlichen Indexstrukturen oder verwandten Techniken

- Indexstrukturen liefern ein grobes Vor-Clustering
  räumlich benachbarte Objekte werden möglichst auf der gleichen Seite abgespeichert
- Indexstrukturen sind effizient
  nur einfache Heuristiken zum Clustering
- schnelle Zugriffsmethoden für verschiedene Ähnlichkeitsanfragen
  z.B. Bereichsanfragen und k-Nächste-Nachbarn-Anfragen

3.5 Indexbasiertes Sampling

Methode [Ester, Kriegel & Xu 1995]

- Aufbau eines R-Baums
- Auswahl von Repräsentanten von den Datenseiten des R-Baums
- Anwendung des Clustering-Verfahrens auf die Repräsentantenmenge
- Übertragung des Clustering auf die gesamte Datenbank



Datenseitenstruktur eines R*-Baums

3.5 Indexbasiertes Sampling

Übertragung des Clustering auf die Grundgesamtheit

- Wie erhält man aus dem Clustering der Stichprobe ein Clustering der Grundgesamtheit?
- Bei *k*-means- und *k*-medoid-Verfahren:

  Repräsentanten der Cluster für die gesamte Datenmenge übernehmen (Centroide, Medoide)
- Bei dichtebasierten Verfahren:

  eine Repräsentation für jedes Cluster bilden (z.B. MUR)

  die Objekte dem „nächsten“ der gefundenen Cluster zuweisen
- Bei hierarchischen Verfahren:

  ⇨ Generierung einer hierarchischen Repräsentation problematisch!

  (Dendrogramm oder Erreichbarkeits-Diagramm)

3.5 Indexbasiertes Sampling

Auswahl von Repräsentanten

Wieviele Objekte sollen von jeder Datenseite ausgewählt werden?

- hängt vom verwendeten Clusteringverfahren ab
- hängt von der Verteilung der Daten ab
- z.B. für CLARANS: ein Objekt pro Datenseite

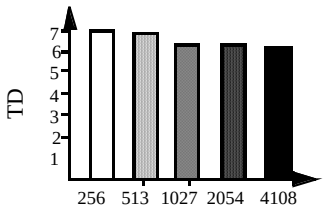
guter Kompromiß zwischen der Qualität des Clusterings und der Laufzeit

Welche Objekte sollen ausgewählt werden?

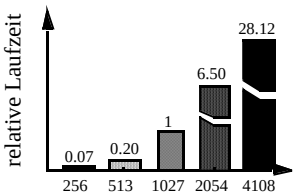
- hängt ebenfalls vom Clusteringverfahren und von der Verteilung der Daten ab
- einfache Heuristik: wähle das „zentralste“ Objekt auf der Datenseite

3.5 Indexbasiertes Sampling

Experimentelle Untersuchung für CLARANS



Anzahl der Repräsentanten



Anzahl der Repräsentanten

- Laufzeit von CLARANS ist etwa $O(n^2)$
- Qualität des Clusterings steigt bei mehr als 1024 Repräsentanten kaum noch

⇨ 1024 Repräsentanten guter Kompromiß zwischen Qualität und Effizienz

3.5 Bereichsanfragen für dichtebasiertes Clustering

- Basisoperation für DBSCAN und für OPTICS:
  Berechnung der ε-Nachbarschaft jedes Objekts *o* in der Datenbank
- effiziente Unterstützung von Bereichsanfragen durch räumliche Indexstrukturen

  R-Baum, X-Baum, M-Baum, . . .
- Laufzeitkomplexitäten für die Algorithmen DBSCAN und OPTICS:

	einzelne Bereichsanfrage	gesamter Algorithmus
ohne Index	$O(n)$	$O(n^2)$
mit Index	$O(\log n)$	$O(n \log n)$
mit direktem Zugriff	$O(1)$	$O(n)$

Probleme räumlicher Indexstrukturen bei hochdimensionalen Daten

3.5 GRID-Clustering

Methode [Schikuta 1996]

Grob-Clustering durch räumliche Indexstruktur

das Volumen des durch eine Datenseite repräsentierten Datenraums ist um so kleiner, je höher die Punktdichte in diesem Gebiet des Raums ist

Nachbearbeitung durch Verschmelzen von Seitenregionen

Seitenregionen mit hoher Punktdichte werden als Clusterzentren angesehen und rekursiv mit benachbarten, weniger dichten Seitenregionen verschmolzen

⇒ dichtebasiertes Clustering

Verwendete Indexstruktur

Gridfile

3.5 GRID-Clustering

Methode

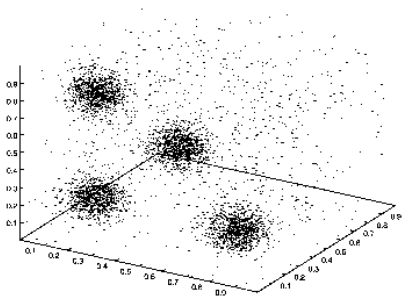
- beginne mit der Datenseite *S*, die die höchste Punktdichte hat
- die Seite *S* wird dann mit allen (direkten und indirekten) Nachbarseiten *R* verschmolzen, deren Punktdichte kleiner oder gleich der Punktdichte von *S* ist
- wenn es nur noch Nachbarseiten mit höherer Punktdichte gibt:

beginne einen neuen Cluster mit der Seite, die nun die höchste Punktdichte unter den noch nicht betrachteten Datenseiten hat

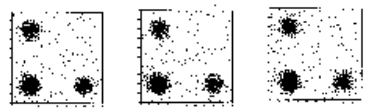
mit der zusätzlichen Information über die Verschmelzungsreihenfolge läßt sich das Ergebnis des Algorithmus als Dendrogramm darstellen!

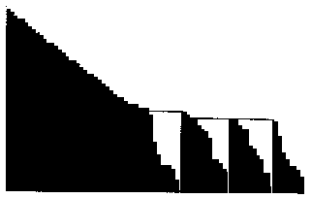
3.5 GRID-Clustering

Beispiel



3-dimensionale Punktdaten





Resultierendes Dendrogramm

3.5 Datenkompression zum Vor-Clustering

Grundlagen [Zhang, Ramakrishnan & Linvy 1996]

Methode

- Bildung kompakter Beschreibungen von Teil-Clustern (Clustering Features)
- hierarchische Organisation der Clustering Features
  in einem höhenbalancierten Baum (CF-Baum)
- Anwendung eines Clusteringverfahren wie z.B. CLARANS
  auf die Blätter des Baums

CF-Baum

- komprimierte, hierarchische Repräsentation der Daten
- berücksichtigt die Clusterstruktur

3.5 Datenkompression zum Vor-Clustering

Grundbegriffe

Clustering Feature einer Menge C von Punkten X_i: $CF = (N, LS, SS)$

$N = |C|$ „Anzahl der Punkte in C“

$LS = \sum_{i=1}^{N} X_i$ „lineare Summe der N Datenpunkte“

$SS = \sum_{i=1}^{N} X_i^2$ „Quadratsumme der N Datenpunkte“

aus den CF‘s können berechnet werden

- Centroid
- Kompaktheitsmaße
- und Distanzmaße für Cluster

3.5 Datenkompression zum Vor-Clustering

Grundbegriffe

- Additivitätstheorem

  für CF-Vektoren für zwei disjunkte Cluster C_1 und C_2 gilt

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

  d.h. CF‘s können inkrementell berechnet werden

- Definition

  Ein *CF-Baum* ist ein höhenbalancierter Baum zur Abspeicherung von CF‘s.

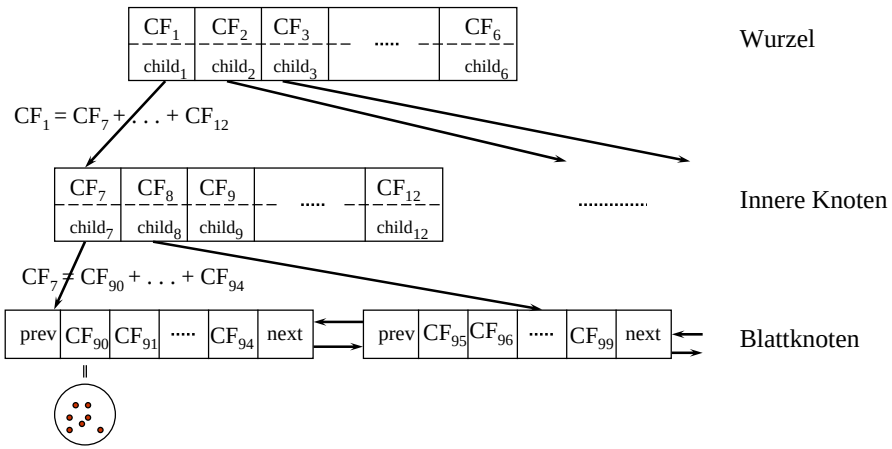
3.5 Datenkompression zum Vor-Clustering

Grundbegriffe

- Eigenschaften eines CF-Baums:
  - Jeder innere Knoten enthält höchstens B Einträge der Form $[CF_i, child_i]$ und CF_i ist der CF-Vektor des Subclusters des i-ten Sohnknotens.
  - Ein Blattknoten enthält höchstens L Einträge der Form $[CF_i]$.
  - Jeder Blattknoten besitzt zwei Zeiger *prev* und *next*.
  - Der Durchmesser aller Einträge in einem Blattknoten ist kleiner als T
- Aufbau eines CF-Baums
  - Transformation eines Datensatzes p in einen CF-Vektor $CF_p=(1, p, p^2)$
  - Einfügen von CF_p analog dem Einfügen in einen B^+-Baum
  - bei Verletzung des Schwellwertes T wird das entsprechende Blatt gesplittet

Beispiel

B = 7, L = 5



BIRCH

Phase 1

- ein Scan über die gesamte Datenbank
- Aufbau eines CF-Baums B_1 bezgl. T_1 durch sukzessives Einfügen der Datensätze

Phase 2

- falls der CF-Baum B_1 noch zu groß ist, wähle ein $T_2 > T_1$
- Aufbau eines CF-Baums B_2 bzgl. T_2 durch Einfügen der CF‘s der Blätter von B_1

Phase 3

- Anwendung eines Clusteringalgorithmus auf die Blatteinträge des CF-Baums
- Clusteringalgorithmus (z.B. CLARANS) muss evtl. an Clustering Features angepaßt werden

Diskussion

+ Komprimierungsfaktor frei wählbar
+ Effizienz

  Aufbau eines sekundärspeicherresidenten CF-Baums: O(n log n)

  Aufbau eines hauptspeicherresidenten CF-Baums: O(n)

  zusätzlich: Aufwand des Clusteringalgorithmus

- nur für numerische Daten
  euklidischer Vektorraum
- abhängig von der Reihenfolge der Daten

Überblick

- kategorische Attribute

  „modes“ statt means als Repräsentanten
- ausgedehnte Objekte

  verallgemeinertes dichtebasiertes Clustering
- kontinuierliche Updates der Datenbank

  inkrementelles Clustering
- Cluster nur in Unterräumen des Datenraums

  Subspace Clustering

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

$m = (m_1, ..., m_d)$, *dist* eine Distanzfunktion für kategorische Attribute, z.B.

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

3.6 Clustering mit kategorischen Attributen

Bestimmung des Modes

- Die Funktion $TD(C,m) = \sum_{p \in C} dist(p,m)$ wird minimiert genau dann, 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

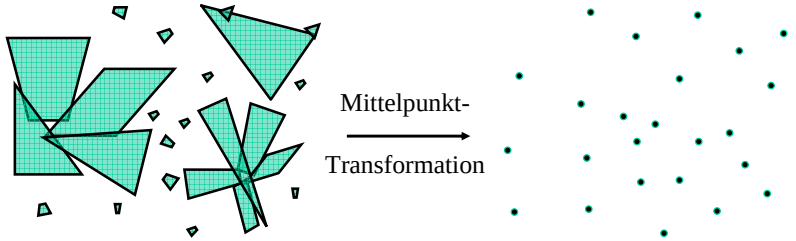
3.6 Clustering mit kategorischen Attributen

Algorithmus k-modes

- Initialisierung
  nicht zufällig
  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

3.6 Clustering ausgedehnter Objekte

Grundlagen



Berücksichtigung der Fläche und nicht-räumlicher Attribute
natürlicher Begriff der Verbundenheit

3.6 Clustering ausgedehnter Objekte

Verallgemeinertes dichte-basiertes Clustering

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

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

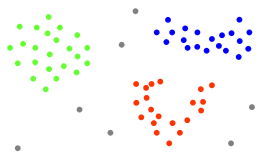
MinWeight(N)
beliebiges Prädikat für
Mengen von Objekten

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

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

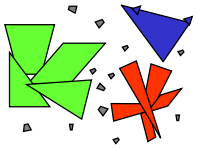
3.6 Clustering ausgedehnter Objekte

Beispiele



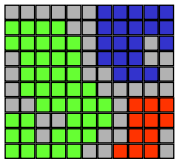
dist(p,q) ≤ ε

cardinality(. . .) ≥ MinPoints



intersect(p,q)

Summe der Flächen ≥
5 % der Gesamtfläche



Nachbarzelle und
ähnliche Farbe

true

3.6 Clustering ausgedehnter Objekte

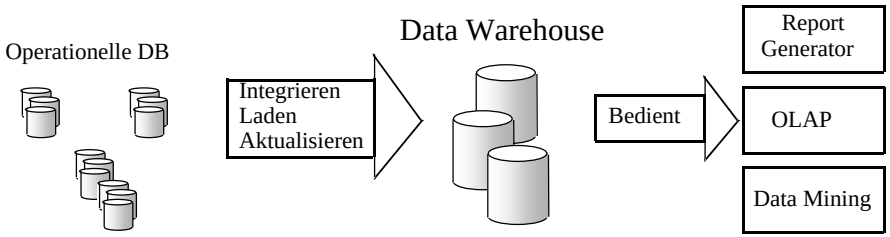
Algorithmus GDBSCAN

- dasselbe algorithmische Schema wie DBSCAN
- anstelle einer N_ε-Anfrage eine N_{NPred}-Anfrage
- anstelle der Bedingung $|N_\varepsilon| \geq MinPts$

  das *MinWeight*-Prädikat auswerten
- Laufzeitkomplexität O(*n* log*n*) bei geeigneter Unterstützung der N_{NPred}-Anfrage

3.6 Inkrementelles dichte-basiertes Clustering

Data Mining in einem Data Warehouse



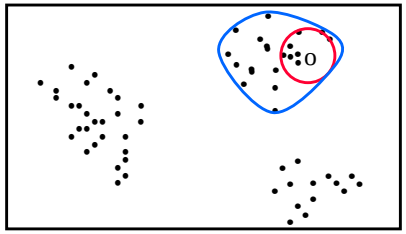
- Updates werden gesammelt und periodisch im Data Warehouse nachgeführt
- alle vom Data Warehouse abgeleiteten Muster müssen aktualisiert werden

⇨ inkrementelle Data-Mining-Algorithmen

Inkrementelles GDBSCAN

[Ester, Kriegel, Sander, Wimmer & Xu 1998]

- nicht die ganze aktualisierte Datenbank erneut clustern
- nur die alten Cluster und die eingefügten / gelöschten Objekte betrachten
- GDBSCAN: nur die Nachbarschaft eines eingefügten / gelöschten Objekts und die davon dichte-erreichbaren Objekte sind *betroffen*



o Einfügung / Löschung

N_{NPred} (o)

Vom Update betroffen

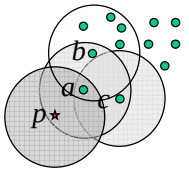
Grundlagen

- MinWeight-Prädikat muß *inkrementell auswertbar* sein

  $weight(N) = \sum_{o \in N} weight(\{o\})$ und *MinWeight*(*N*) definiert als $weight(N) \geq T$
- *Randobjekt*: gehört zum Cluster, ist aber kein Kernobjekt
- Potentielle Konsequenzen der Einfügung oder Löschung eines Objekts *p*

  In $N_{NPred}(p)$: Kernobjekte ↔ Randobjekte ↔ Rauschen

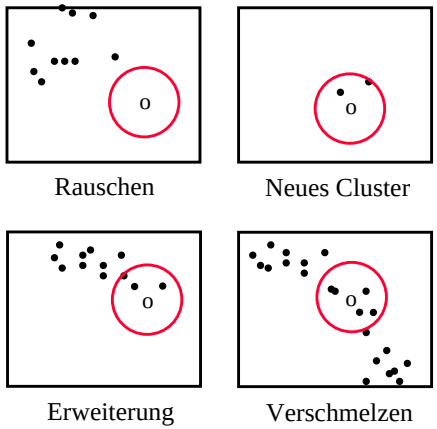
  In $N_{NPred}(q)$ mit $q \in N_{NPred}(p)$: Randobjekte ↔ Rauschen



MinPts = 4, ε wie gezeigt

a: Randobjekt ↔ Kernobjekt
c: Rauschen ↔ Randobjekt

Einfüge-Algorithmus



o Einfügung

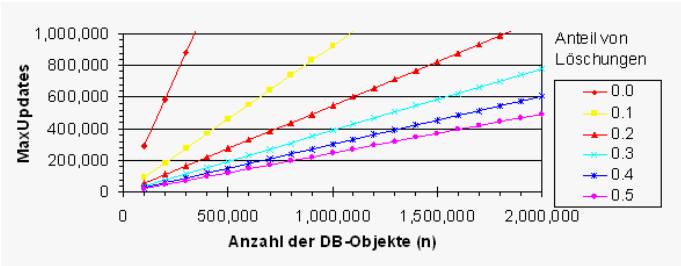
MinPts = 3, ε wie gezeigt

Virtuelle Cluster IDs

- speichere die Information, welche Cluster verschmolzen wurden
- Verschmelzen erfordert keinen Zugriff auf die betroffenen Cluster

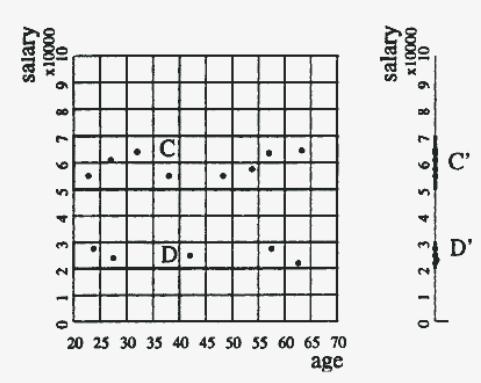
Experimentelle Untersuchung

MaxUpdates: Zahl von Updates, bis zu denen Inkrementelles GDBSCAN effizienter ist als GDBSCAN angewendet auf die ganze aktualisierte Datenbank



⇨ sogar für 50 % Löschungen: *MaxUpdates* = 25 % der Datenbank Größe

Motivation



Cluster nur im
1-dimensionalen Unterraum
„salary“

CLIQUE [Agrawal, Gehrke, Gunopulos & Raghavan 1998]

1. Identifikation von Unterräumen mit Clustern
2. Identifikation von Clustern
3. Erzeugung von Cluster-Beschreibungen

- *Cluster*: „dichte Region“ im Datenraum
- Dichte-Grenzwert τ

  Region ist *dicht*, wenn sie mehr als τ Punkte enthält
- Gitterbasierter Ansatz

  jede Dimension wird in ξ Intervalle aufgeteilt

  Cluster ist Vereinigung von verbundenen dichten Regionen

Identifikation von Unterräumen mit Clustern

- Aufgabe: Entdecken dichter Basis-Regionen
- naiver Ansatz

  berechne Histogramme für alle Teilmengen der Dimensionen

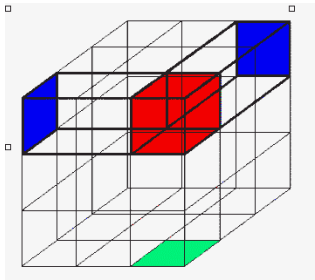
  ⇨ ineffizient für hoch-dimensionale Daten (O (2^d) für *d* Dimensionen)
- Greedy-Algorithmus (Bottom-Up)

  beginne mit der leeren Menge

  nehme jeweils eine Dimension dazu
- Grundlage dieses Algorithmus: *Monotonie-Eigenschaft*

  wenn eine Region *R* im *k*-dimensionalen Raum dicht ist, dann ist auch jede Projektion von *R* in einen (*k*-1)-dimensionalen Unterraum dicht

Beispiel



2-dim. dichte Regionen

3-dim. Kandidaten-Region

2-dim. Region, die geprüft werden muß

- Laufzeitkomplexität des Greedy-Algorithmus $O(\varsigma^k + n \cdot k)$

  für *n* Datenbank-Objekte und *k* = höchste Dimension einer dichten Region
- heuristische Reduktion der Anzahl der Kandidaten-Regionen

  Anwendung des „Minimum Description Length“ - Prinzips

Identifikation von Clustern

- Aufgabe: Finden maximaler Mengen verbundener dichter Regionen
- Gegeben: alle dichten Regionen in demselben *k*-dimensionalen Unterraum
- „depth-first“-Suche in folgendem Graphen (Suchraum)

  Knoten: dichte Regionen

  Kanten: gemeinsame Kanten / Dimensionen der beiden dichten Regionen
- Laufzeitkomplexität

  dichte Regionen im Hauptspeicher (z.B. Hashbaum)

  für jede dichte Region 2 *k* Nachbarn zu prüfen

  $\Rightarrow$ Zahl der Zugriffe zur Datenstruktur: 2 *k n*

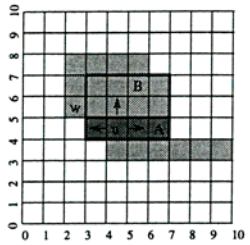
Erzeugung von Cluster-Beschreibungen

- Gegeben: ein Cluster, d.h. eine Menge verbundener dichter Regionen
- Aufgabe: Finden optimaler Überdeckung dieses Clusters durch eine

  Menge von Hyperrechtecken
- Standard-Methoden

  das Problem ist NP-hart

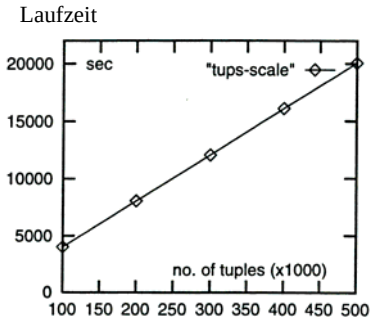
  zu ineffizient für große Werte von *d*

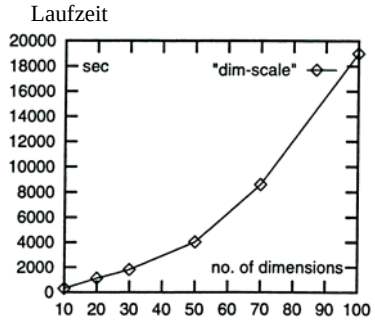


- Heuristische Methode

  1. überdecke das Cluster durch maximale Regionen
  2. entferne redundante Regionen

Experimentelle Untersuchung





Laufzeitkomplexität von CLIQUE

linear in *n* , superlinear in *d*

Diskussion

+ automatische Entdeckung von Unterräumen mit Clustern
+ automatische Entdeckung von Clustern
+ keine Annahme über die Verteilung der Daten
+ Unabhängigkeit von der Reihenfolge der Daten
+ gute Skalierbarkeit mit der Anzahl *n* der Datensätze

- Genauigkeit des Ergebnisses hängt vom Parameter ξ ab
- braucht eine Heuristik, um den Suchraum aller Teilmengen der Dimensionen einzuschränken

  $\Rightarrow$ findet u.U. nicht alle Unterräume mit Clustern