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

Skript zur Vorlesung

Knowledge Discovery in Databases

im Wintersemester 2009/2010

Kapitel 8: DB-Techniken zur Leistungssteigerung

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

8. DB-Techniken zur Leistungssteigerung

Bisher: (meist) 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)

Übersicht über das Kapitel

➡ Hauptaugenmerk auf Clustering Algorithmen

8.1 Datenkompression

Idee:

- DM-Algorithmus auf der gesamten Datenmenge zu teuer
- Komprimiere Daten, so dass sie in den Hauptspeicher passen
- Wende DM-Algorithmus auf komprimierte Daten an

Techniken:

- Sampling (8.1.1)
  Datenbank wird auf eine Stichprobe reduziert

- Micro-Clustering (8.1.2)
  bilde Micro-Cluster, die Teilmengen der Daten möglichst genau repräsentieren; Datenbank wird auf Micro-Cluster reduziert

8.1.1 Sampling

Indexbasiertes Sampling [Ester, Kriegel & Xu 1995]

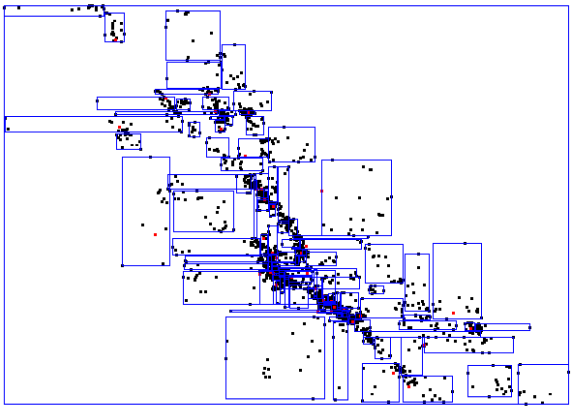
Zufälliges Sampling liefert u.U. schlechte Qualität

Verwendung von räumlichen Indexstrukturen oder verwandten Techniken zur Auswahl des Samplings

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

Methode

- 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

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

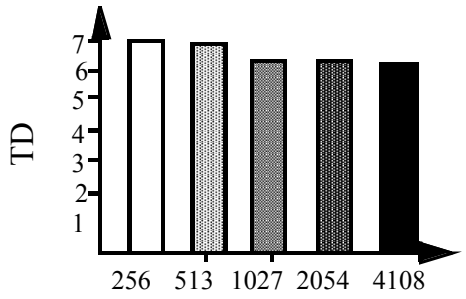
guter Kompromiss zwischen der Qualität des Clusterings und der Laufzeit

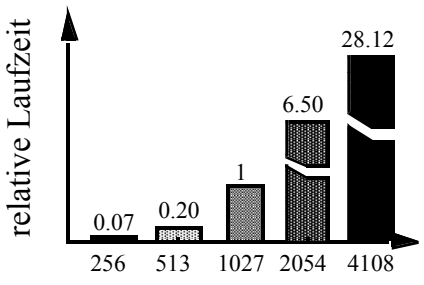
Welche Objekte sollen ausgewählt werden?

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

8.1.1 Sampling

Experimentelle Untersuchung für CLARANS





- 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

8.1.2 Micro-Clustering

BIRCH [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 Clustering Features in den Blättern des Baums

CF-Baum

- komprimierte, hierarchische Repräsentation der Daten
- berücksichtigt die Clusterstruktur

8.1.2 Micro-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“

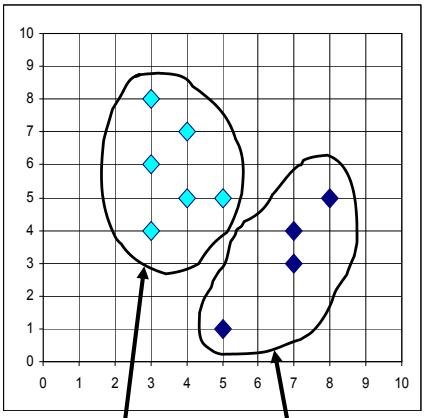
aus den CF's können berechnet werden

- Centroid (Repräsentant)
- Kompaktheitsmaße
- und Distanzmaße für Cluster

8.1.2 Micro-Clustering

Beispiel

(3,4)
(4,5)
(5,5)
(3,6)
(4,7)
(3,8)



(5,1)
(7,3)
(7,4)
(8,5)

$CF_1 = (6, (22,35),(84,215))$

$CF_2 = (4, (27,13),(187,51))$

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

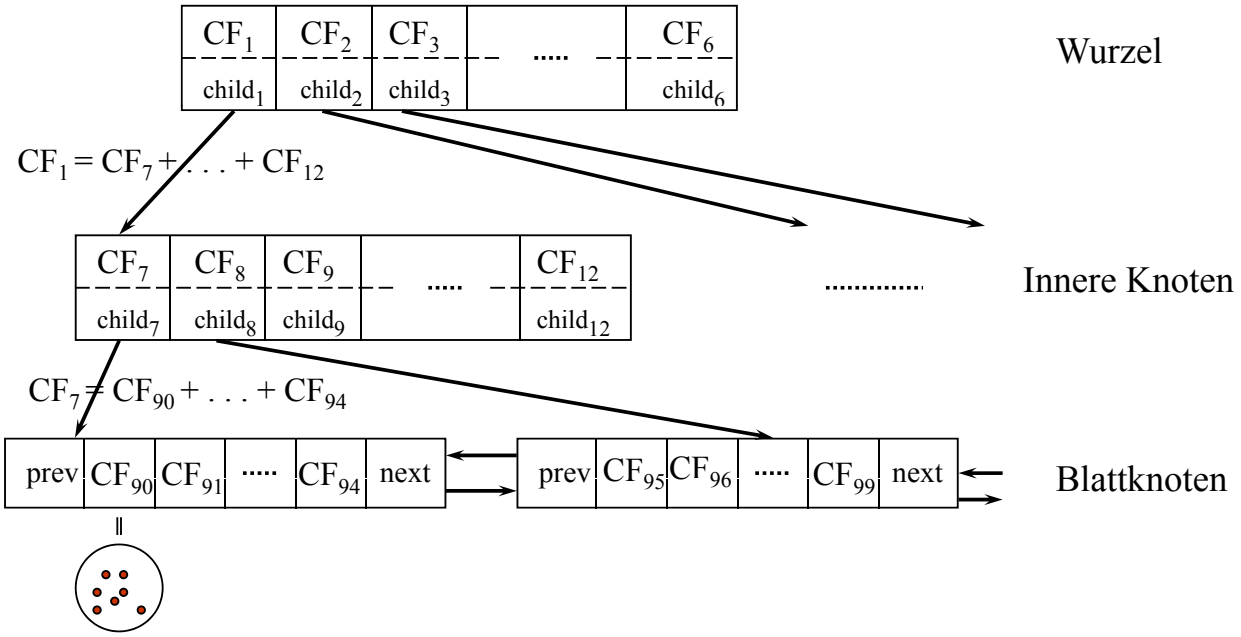
  Ein *CF-Baum* ist ein höhenbalancierter Baum zur Abspeicherung von CF's.

- **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*.
  - Für jeden Eintrag eines Blattknotens ist der Durchmesser kleiner als T.
- **Aufbau eines CF-Baums (analog B⁺-Baum)**
  - Transformation eines Datensatzes p in einen CF-Vektor $CF_p=(1, p, p^2)$
  - Einfügen von CF_p in den Teilbaum des CF-Vektors mit kleinster Distanz
  - bei Verletzung des Schwellwerts T wird ein neuer Eintrag CF_p eingefügt, sonst absorbiert der nächste Eintrag im Blatt CF_p
  - bei Verletzung des Schwellenwerts B oder L wird Knoten gesplittet:
    - die entferntesten CF's bilden die beiden neuen Knoten
    - die restlichen CF's werden dem neuen Knoten mit geringster Distanz zugeordnet

8.1.2 Micro-Clustering

Beispiel

B = 7, L = 5



8.1.2 Micro-Clustering

Verfahren

Phase 1

- ein Scan über die gesamte Datenbank
- Aufbau eines CF-Baums B_1 bzgl. 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 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)

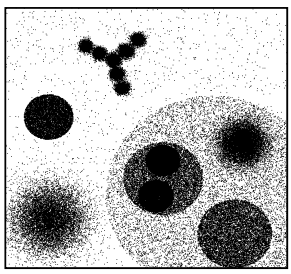
zusätzlich: Aufwand des Clusteringalgorithmus
(wenn CF-Baum im Hauptspeicher, ist dieser Aufwand vernachlässigbar)

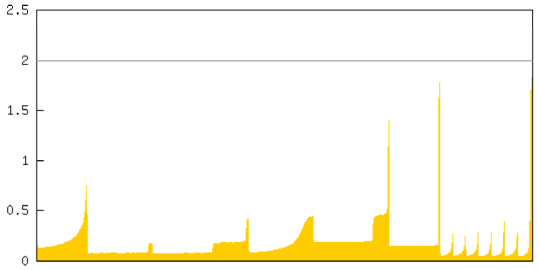
– nur für numerische Daten (euklidischer Vektorraum)
– abhängig von der Reihenfolge der Daten

Data Bubbles [Breunig, Kriegel, Kröger, Sander 2001]

Original DB und OPTICS-Plot

1 Mio. Datenpunkte

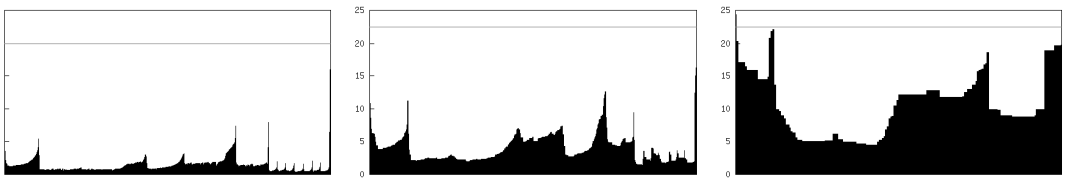




OPTICS-Plots auf Komprimierten Daten

k=10,000 k=1,000 k=200

Sampling

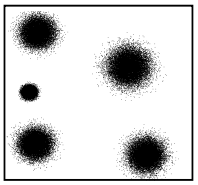


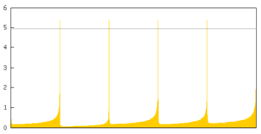
BIRCH

8.1.2 Micro-Clustering

Drei Hauptprobleme bei BIRCH und Sampling für hierarchisches Clustering:

1. Verlorengegangene Objekte (Lost Objects)
   - Viele Objekte der DB fehlen im Plot/Dendrogram
2. Größenverzerrung (Size Distortions)
   - Cluster sind gequetscht und gestreckt
3. Strukturelle Verzerrung (Structural Distortions)
   - hierarchische Cluster Struktur ist zerstört

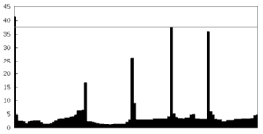




OPTICS original

Lösungen

- Post-Processing für Problem 1 und 2 (Lost Objects, Size Distortions)
  - nn-Klassifikation, ersetze repräsentative Objekte durch die Menge der repräsentierten Objekte
- Data Bubbles für Problem 3 (Structural Distortions)



Sampling 100 Obj.

CF 100 Obj.

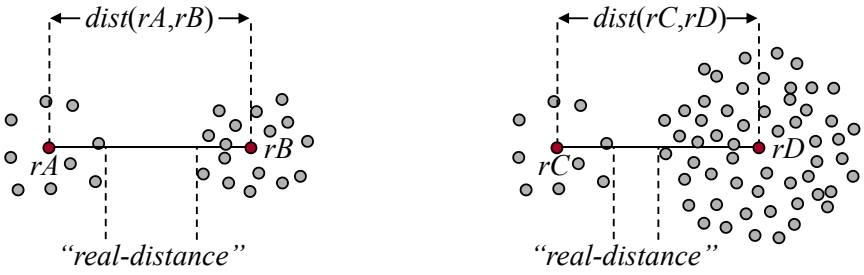
DATABASE SYSTEMS GROUP

8.1.2 Micro-Clustering

LMU

Gründe für strukturelle Verzerrungen:

- Distanz zwischen den Originalobjekten wird schlecht durch die Distanz zwischen Repräsentanten approximiert



- Erreichbarkeitsdistanz die den Repräsentanten zugeordnet werden, approximieren die Erreichbarkeitsdistanz der repräsentierten Objekte sehr schlecht

8.1.2 Micro-Clustering

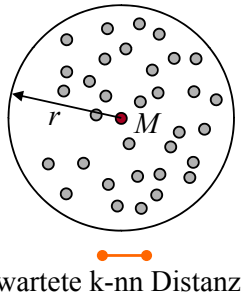
Data Bubble: reichere Abstraktion

Definition: **Data Bubble**

- Data Bubble $B=(n, M, r)$ für eine Menge von n Objekten $X=\{X_i\}$

$M = \left(\sum_{i=1}^{n} X_i \right) / n$ ist der *Mittelpunkt* von X und

$r = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(X_i - X_j)^2}{n \cdot (n-1)}}$ ist der *Radius* von X.



- Erwartete k-nn Distanz der Objekte X_i im Data Bubble (bei Gleichverteilung)

$$nnDist(k, B) = r \cdot \left(\frac{k}{n} \right)^d$$

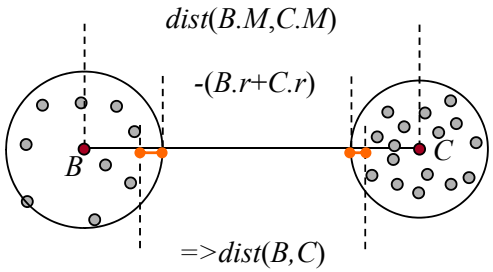
- Data Bubbles können entweder aus einem Sample oder aus CFs berechnet werden

DATABASE SYSTEMS GROUP

8.1.2 Micro-Clustering

LMU

Definition: Distanz zwischen Data Bubbles



Definition: Kern- und Erreichbarkeitsdistanz für Data Bubbles

- analog zur Kern- und Erreichbarkeitsdistanz von Punkten

Definition: virtuelle Erreichbarkeitsdistanz eines Data Bubble

- Erwartete k-nn-Distanz innerhalb des Data Bubble
- bessere Approximation der Erreichbarkeitsdistanz der repräsentierten Objekte

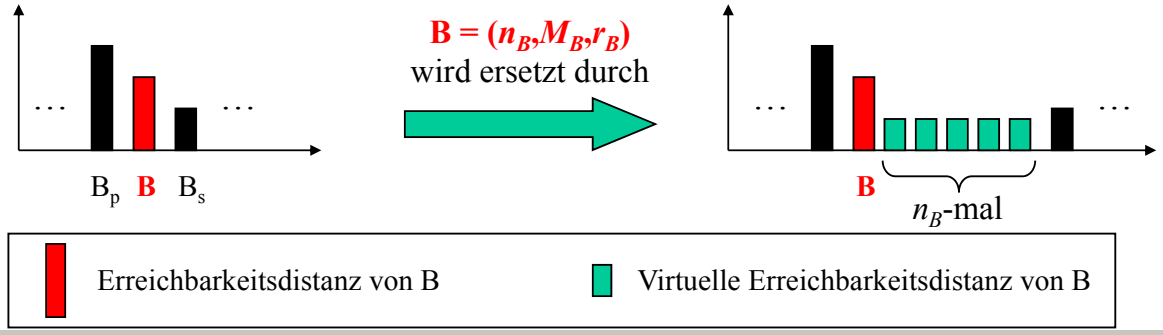
8.1.2 Micro-Clustering

Clustering mit Data Bubbles:

- Generiere *m* Data Bubbles aus *m* Sampleobjekten oder CF-Features
- Cluster die Menge der Data Bubbles mit OPTICS
- Generiere das Erreichbarkeitsdiagramm:

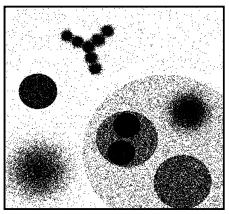
  Für jedes Data Bubble *B*:

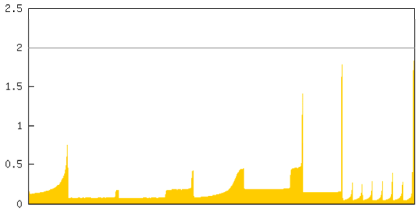
  - „Plotte“ die Erreichbarkeitsdistanz *B.reach* (vom OPTICS-Lauf auf den Data Bubbles erzeugt)
  - „Plotte“ alle Punkte, die von *B* repräsentiert werden mit der virtuellen Erreichbarkeitsdistanz von *B*

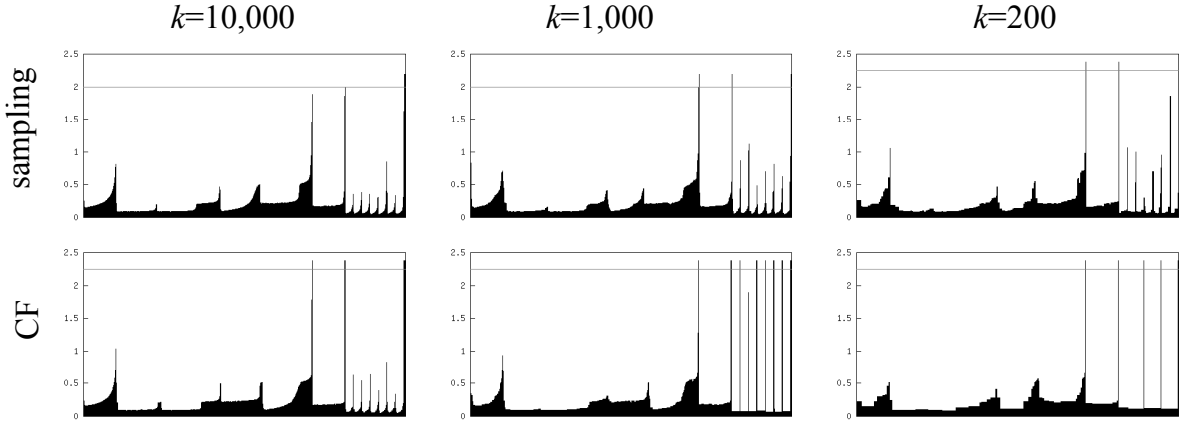


8.1.2 Micro-Clustering

Ergebnisse (Kompression auf *k* Objekte)

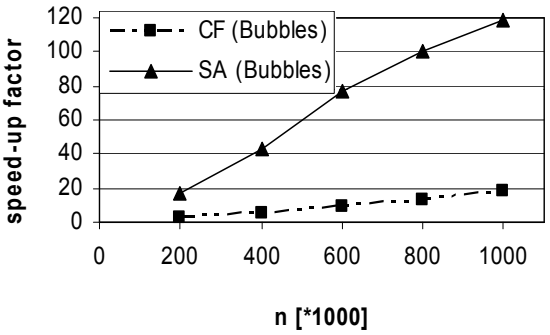






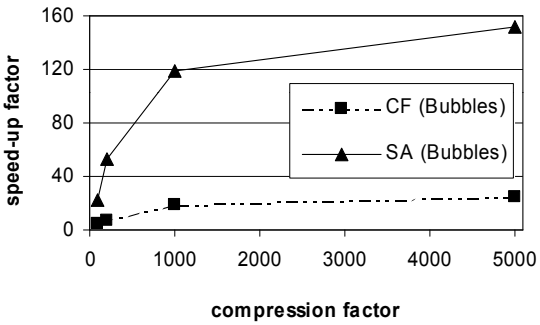
8.1.2 Micro-Clustering

Speed-Up-Faktoren



bzgl. Datenbank Größe

Speed-Up-Faktoren



bzgl. Kompressions Faktor
für Datenbank mit (1 Mio. Objekte)

DATABASE SYSTEMS GROUP

8.1.2 Micro-Clustering

LMU

Diskussion:

- Löst die Probleme Lost Objects, Size Distortions und Structual Distortions
- Data Bubbles aus zufällig gewählten Reerenzobjekten (Sample) erzielt exprimentell bessere Ergebnisse als mit BIRCH erzeugte Data Bubbles
- Nur auf Feature-Räume anwendbar

Erweiterung für allgemein metrische Objekte [Zhou, Sander 2003]

- Repräsentant *M*: Medoid statt Mittelwert
- Distanz zwischen Data Bubbles deutlich komplizierter, da die Berechnung der paarweisen *k*-nn-Distanzen komplex ist

8.1.2 Micro-Clustering

GRID-Clustering [Schikuta 1996]

Idee: 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
  ➡ Micro-Cluster ≡ Datenseite der Indexstruktur

- Verwendete Indexstruktur: *Gridfile*

DATABASE SYSTEMS GROUP

8.1.2 Micro-Clustering

LMU

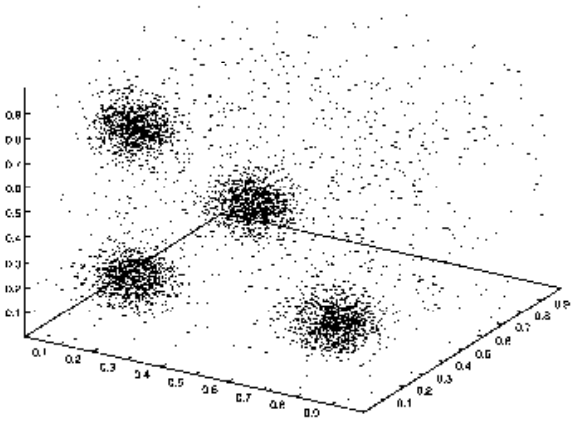
Methode

- 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

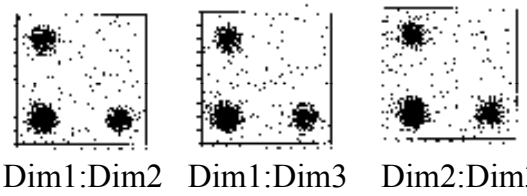
- wenn es nur noch Nachbarseiten mit höherer Punktdichte gibt:
  beginne einen neuen Cluster mit der Seite, die nun die höchste Punktdichte unter den noch nicht betrachteten Datenseiten hat

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

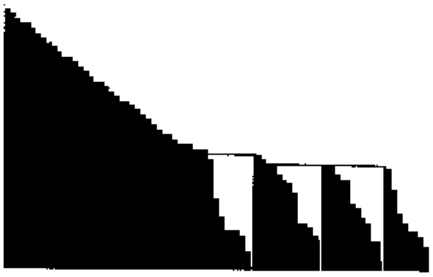
Beispiel



3-dimensionale Punktdaten



Dim1:Dim2 Dim1:Dim3 Dim2:Dim3



Resultierendes Dendrogramm

Fazit Micro-Clustering-Verfahren

- BIRCH gut geeignet für partitionierende Verfahren
- Data Bubbles geeignet für hierarchische Verfahren (speziell Single-Link und OPTICS)
- GridClustering geeignet für hierarchische Verfahren (speziell Single-Link und OPTICS)
- Die Qualität der gefundenen Cluster(-Struktur) hängt bei allen Verfahren ab von der Güte der Microcluster und daher indirekt von der Kompressionsrate
  - D.h. Anzahl der Punkte pro CF-Blatt (bzw. Die Wahl der Schwellwerte B,L und T)
  - Von der Anzahl der Punkte pro Data Bubble
  - Größe der Datenseite der verwendeten Indexstruktur beim Grid-Clustering

8.2 Online Data Mining

Online Data Mining:

Anpassen der Muster (Cluster, Patterns, Klassenmodelle) bei Miteinbeziehen neuer Datenobjekte.

Bespiel:

naive Bayes

- getrenntes Abspeichern von absoluten Häufigkeiten und der Anzahl der Beispiele => relative Häufigkeiten durch inkrement beider Größen
- Naive Bayes ist sehr gut geeignet für schnelles Einbeziehen neuer Daten. Daher häufige Verwendung in Spam-Filtern

K-Means:

- Zuordnen der neuen Objekte zu Centroiden und Update der Centroide
- Danach durchlaufe alle Punkte, wie beim normalen k-Means bis keine Neuzuordnung mehr auftritt.

DATABASE SYSTEMS GROUP

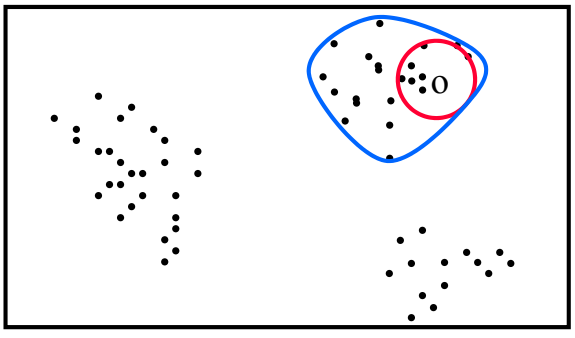
8.2 Online Data Mining

LMU

Inkrementelles DBSCAN

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

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



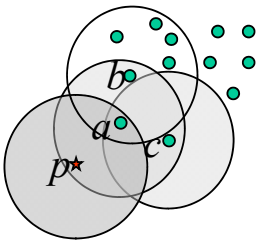
o Einfügung / Löschung

$RQ(o, \varepsilon)$

Vom Update betroffen

Grundlagen

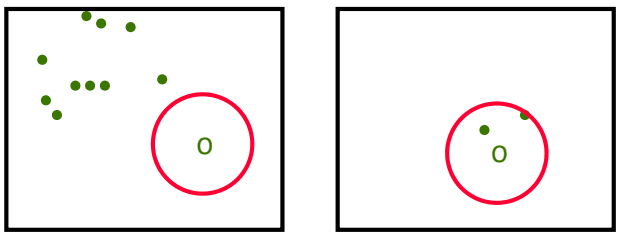
- Kernpunkteigenschaft muss *inkrementell auswertbar* sein
- *Randobjekt*: gehört zum Cluster, ist aber kein Kernobjekt
- Potentielle Konsequenzen der Einfügung oder Löschung eines Objekts *p*
  - In RQ(p,ε): Kernobjekte « Randobjekte « Rauschen
  - In RQ(q,ε) mit q Î RQ(p,ε): Randobjekte « Rauschen



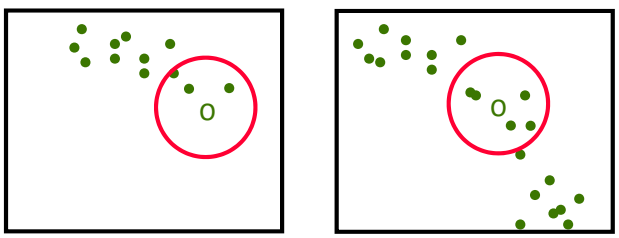
MinPts = 4, ε wie gezeigt

a: Randobjekt ↔ Kernobjekt
c: Rauschen ↔ Randobjekt

Einfügen



Rauschen | Neues Cluster



Erweiterung | Verschmelzen

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

Algorithmus

Einfügen von *p*:

Für alle "neuen" Kernobjekte *o*:
  Verbinde die Objekte in RQ(*o*,ε) mit dem Cluster, dem *o* zugehört
  Wenn Kernobjekte verschiedener Cluster nun miteinander verbunden sind, Verschmelze die entsprechenden Cluster

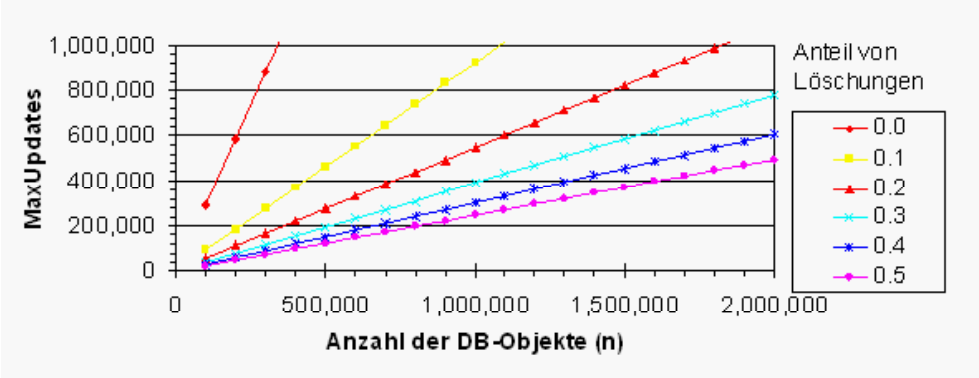
Löschen von *p*:

Für alle "zerstörten" Kernobjekte *o*:
  Setze die Cluster_Id aller Nicht-Kernobjekte aus RQ(*o*,ε) auf Noise;
  Füge alle Kernobjekte aus RQ(*o*,ε) in eine Menge *UpdSeed* ein;
Wende eine Variante von DBSCAN an, die jedoch eine Clusterexpansion nur mit Objekten aus der Menge *UpdSeed* beginnt;

Experimentelle Untersuchung

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

Löschen ist üblicherweise teurer



sogar für 50 % Löschungen/Einfügung (worste-case Verteilung im Realfall): *MaxUpdates* = 25 % der Datenbank Größe