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

Skript zur Vorlesung

Knowledge Discovery in Databases

im Wintersemester 2010/2011

Kapitel 7: DB-Techniken zur Leistungssteigerung

Vorlesung+Übungen:
PD Dr. Peer Kröger, Dr. Arthur Zimek

http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_I_(KDD_I)

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

8.1 Datenkompression
8.2 Online Data Mining

➡ 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

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

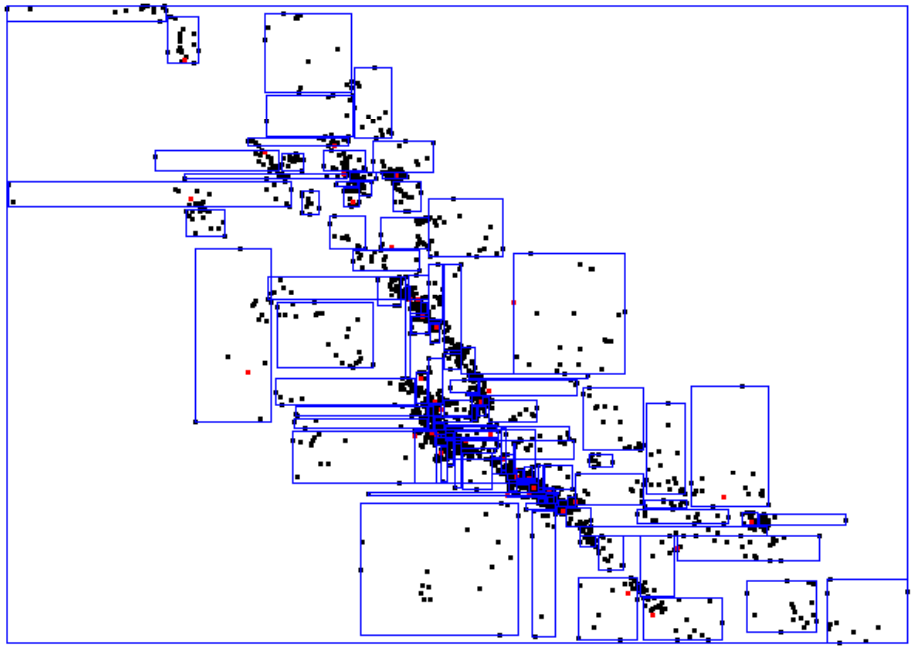
  räumlich benachbarte Objekte werden möglichst auf der gleichen Seite abgespeichert
- Indexstrukturen sind effizient

  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

8.1.2 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“

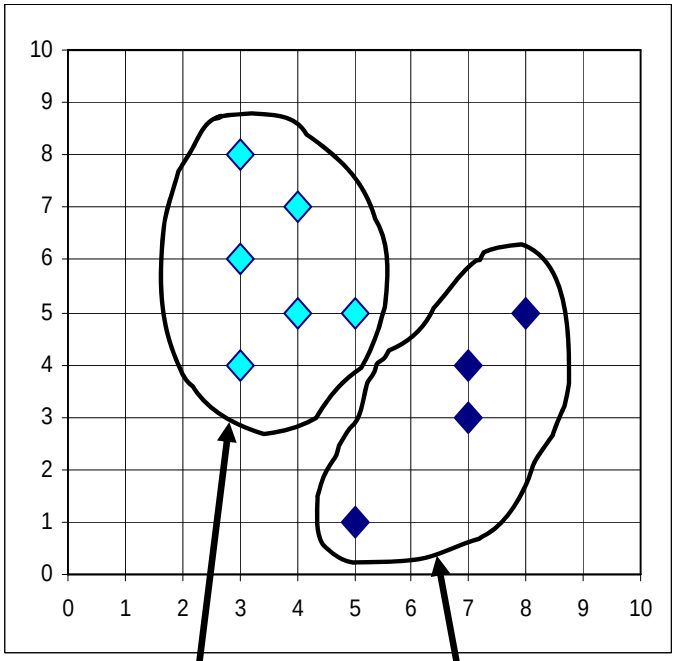
aus den CF‘s können berechnet werden

- 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.

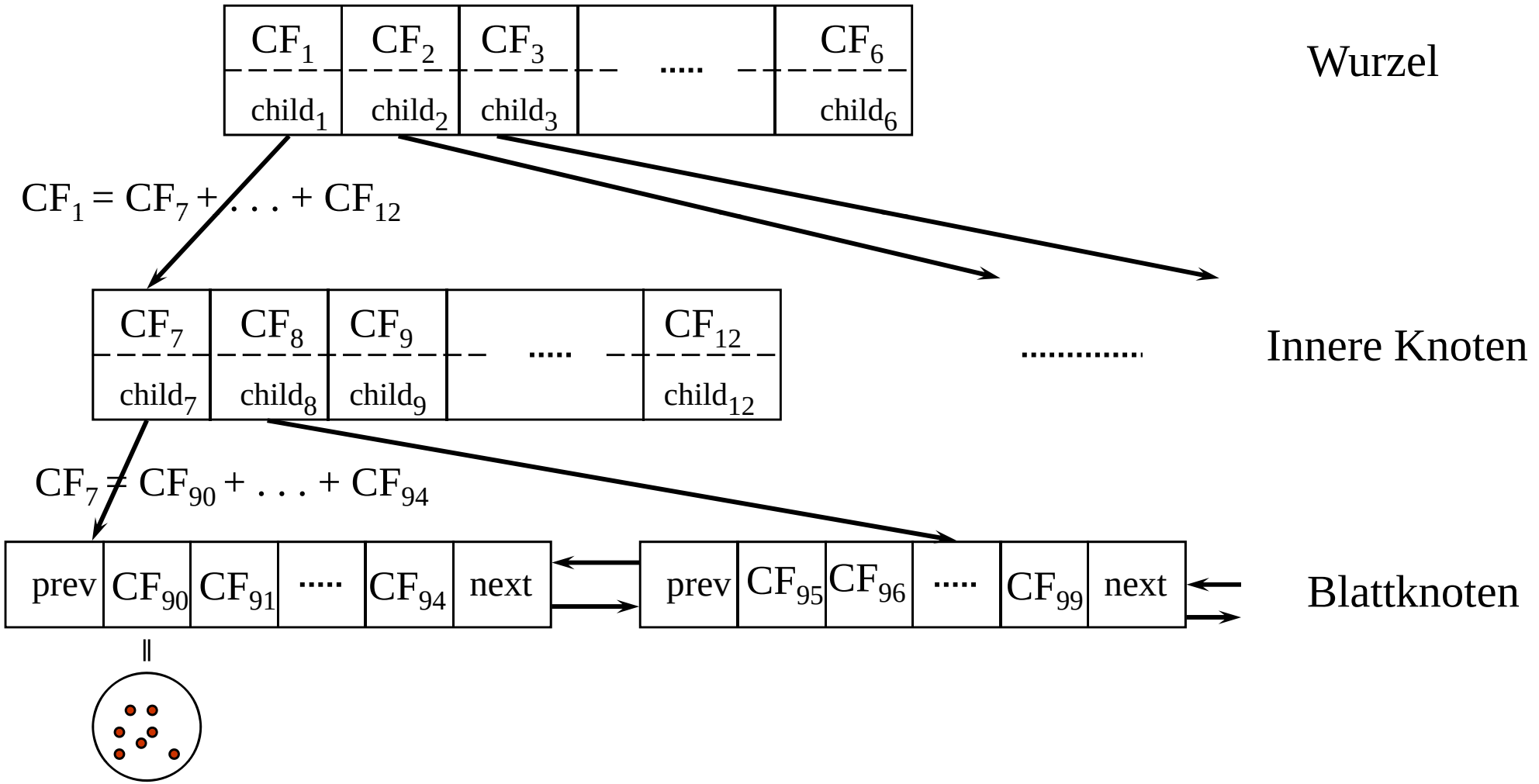
8.1.2 Micro-Clustering

- **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

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

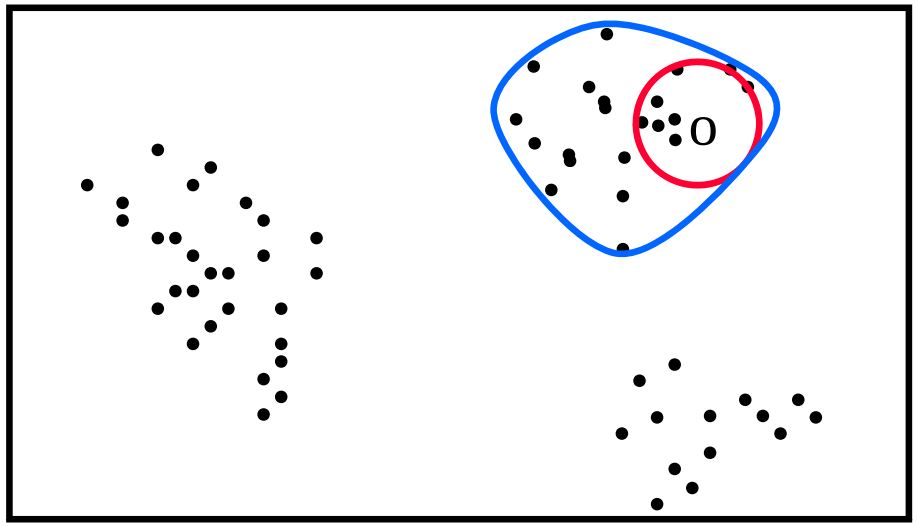
K-Means:

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

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,ε)

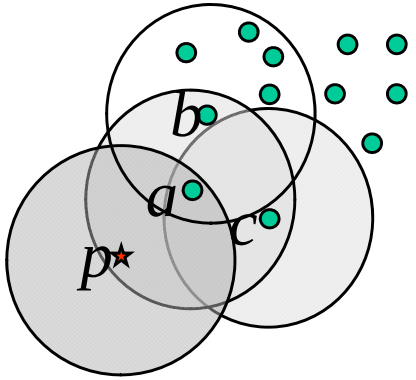
Vom Update betroffen

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

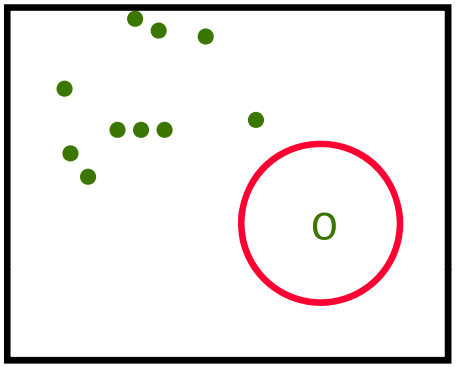


MinPts = 4, ε wie gezeigt

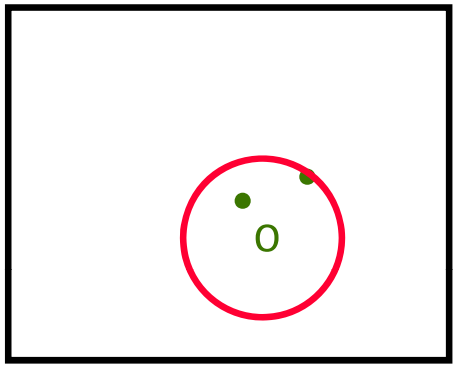
a: Randobjekt ↔ Kernobjekt

c: Rauschen ↔ Randobjekt

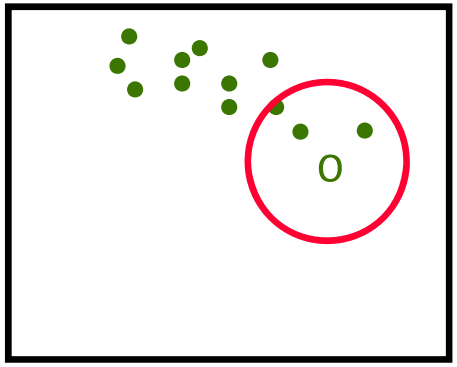
Einfügen



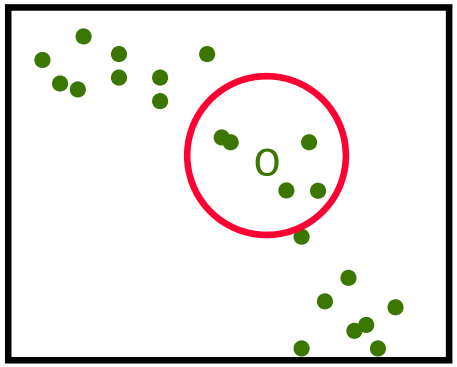
Rauschen



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;