Clustering Data Streams: Theory and Practice

Guha, Meyerson, Mishra, Motwani, O'Callaghan

Philipp Lies

Technische Universität Darmstadt

6. Dezember 2006

Vortragsübersicht

Was ist "Clustering"?

Clustering

Finde eine *Aufteilung* des Datenraums, so dass *ähnliche* Objekte in der gleichen Gruppe sind und verschiedene Objekte in verschiedenen Gruppen sind.

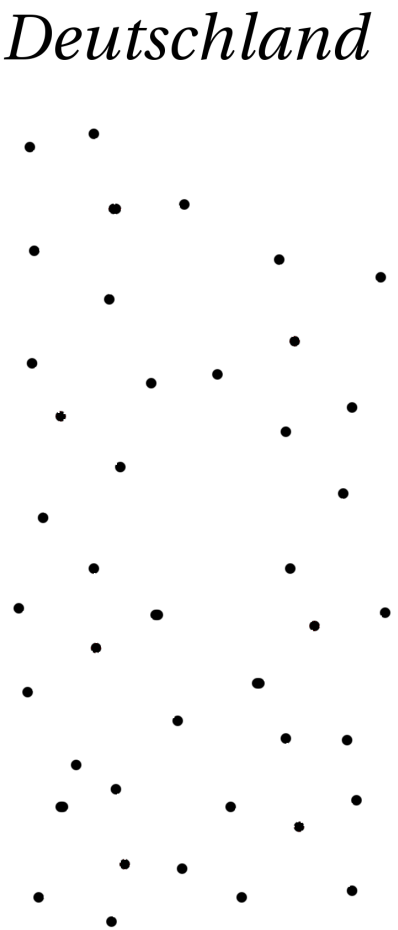
- Auch der Negativeinschluss ist wichtig (vgl *Recall/Precision*) sonst wäre eine Partition mit allen Elementen die ideale Lösung
- Methoden: k-Median, k-Means, k-Center, ...
- Wir betrachten hier: k-Median Clustering

Was ist k-Median? (1)

- k Elemente (Mediane) in einer Menge finden
- Summe der Abstände zwischen jedem Element der Menge und dem nähsten Median soll minimal sein
- Formal: $\sum_{x_i \in N} \min_{c_i \in C} dist(x_i, c_j)$
- k-Mediane einer Menge finden $\Rightarrow$ NP-schweres Problem
- Reduzierung auf Heuristiken notwendig
- Soll nur um konstanten Faktor schlechter sein als Optimum

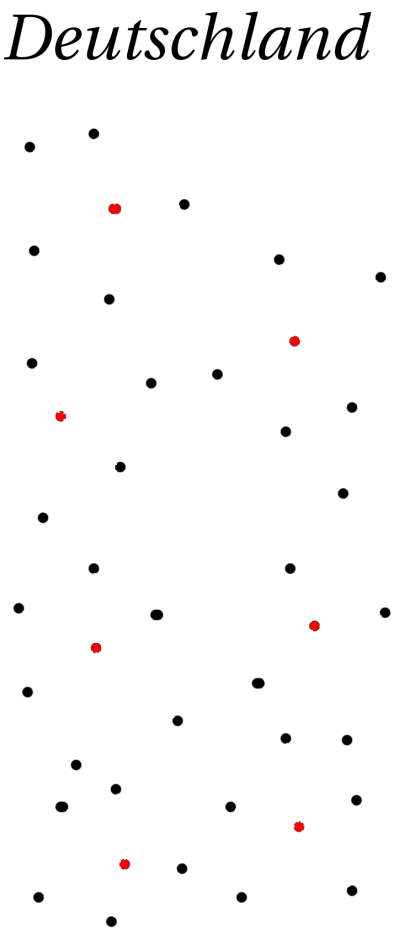
Was ist *k*-Median? (2)

- **Postfilialen** in Deutschland verteilt



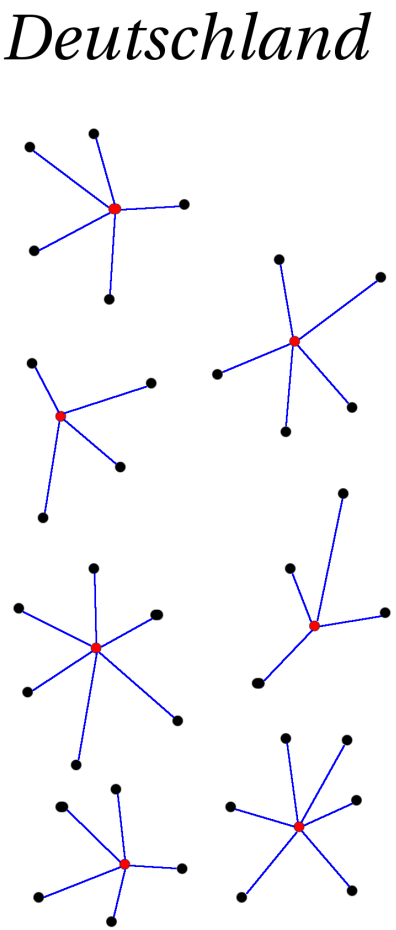
Was ist *k*-Median? (2)

- **Postfilialen** in Deutschland verteilt
- Verteilzentren beliefern umliegende Postfilialen
- *k* Filialen sollen Verteilzentren bekommen



Was ist *k*-Median? (2)

- **Postfilialen** in Deutschland verteilt
- Verteilzentren beliefern umliegende Postfilialen
- *k* Filialen sollen Verteilzentren bekommen
- Lieferweg soll minimal sein
- Sonderfall des *Facility-Location-Problems*

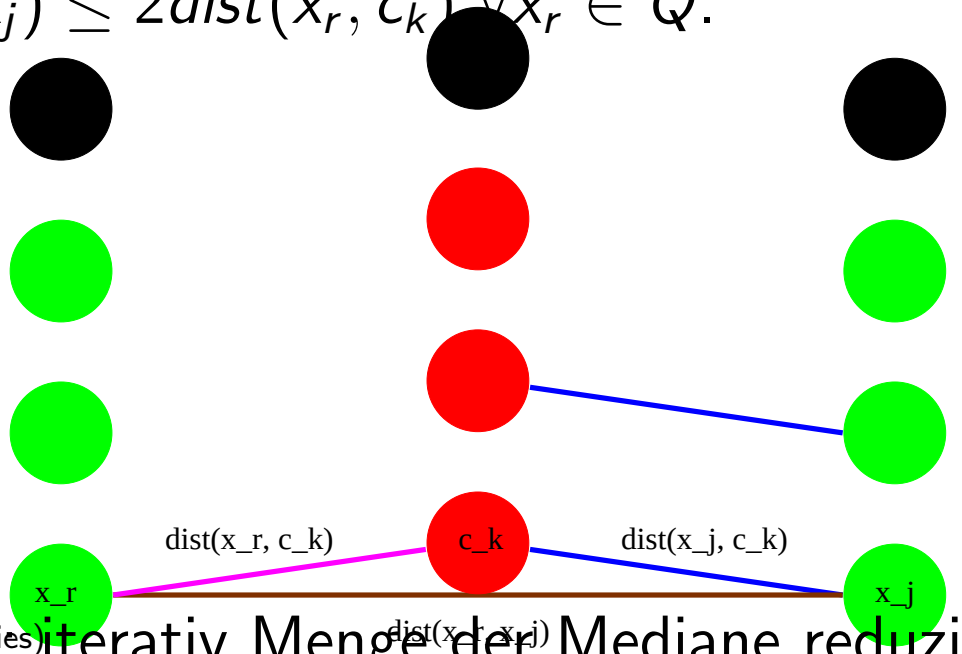


Definitionen

- n_0 Anzahl der Punkte im gesamten Block
- n Anzahl der Eingabepunkte im aktuellen Teil
- M Vorhandener bzw. maximal benutzbarer Speicher
- k Zielanzahl der Mediane für den Algorithmus
- ϵ Konstante $0 \ll \epsilon < 1$, $\epsilon \in \mathbb{R}$
- d Euklidscher Abstand eines Punktes zum nähsten Median
- C Menge der Mediane
- x_i Datenpunkte

Basis

- Vorbedingung:
  Distanzen erfüllen Dreiecksungleichung $|x + y| \leq |x| + |y|$
- Sei S Menge aller Punkte, C_S deren Mediane. Auf reduzierter Menge $Q \subseteq S$ gibt es für jeden Cluster einen Punkt $x_j \in Q$ mit kleinster Distanz zum Median $c_k \in C, \notin Q$. Durch Dreiecksungleichung folgt Abstand $dist(x_r, x_j) \leq 2dist(x_r, c_k) \forall x_r \in Q$.



- Methode iterativ Menge der Mediane reduzieren bis nur

k-Median kurzgefasst

- Der Algorithmus besteht aus 3 Schritten:
  1. $\frac{M}{k}$ Eingabepunkte (Lvl 0) clustern in $2k$ Mediane (Lvl 1)
     - Randomized-Algorithmus
     - Anzahl der zugeordneten Punkte pro Median speichern
  2. M Mediane (Lvl i) clustern in $2k$ Mediane (Lvl $i+1$) mit LSEARCH
  3. Bei $< M$ Mediane clustern in k Mediane mit Primal-Dual-Algorithmus
- Laufzeit: $\tilde{O}(n_o k)$, Speicher: $O(n_0^\epsilon)$

Randomized-Algorithmus

- Nimm $\sqrt{nk}$ zufällig ausgewählte Punkte
- Erzeuge k Mediane mittels Primal-Dual-Algorithmus
- Nimm die $\frac{n}{\sqrt{nk}}$ Punkte mit dem größtem d
- Clustern zu k Medianen mit Primal-Dual-Algorithmus
- $\Rightarrow 2k$ Mediane
- Laufzeit: $O(nk \log nk)$

LSEARCH (1)

- Setze die $z_{\min} = 0$, $z_{\max} = \sum d(x_i, x_0)$, $z = \frac{z_{\max} - z_{\min}}{2}$
- Erzeuge eine zufällige Startlösung
  - Punkte mischen
  - 1. Punkt ist Median
  - Mit Ws(d/z) wird ein Punkt zusätzlicher Median, sonst füge den Punkt zum nähsten Median hinzu
- Extra zufällig $\frac{1}{p} \log k$ Punkte als mögliche Mediane
- $p \in (0, 1)$ untere Schranke für die Größe der Cluster

LSEARCH (2)

solange $|C_{i-1}| \neq k$ wiederhole:

1. Kosten: $cost_i = z|C| + \sum_{i=0}^{n} \min_{c_j \in C} (dist(x_i, c_j))$
2. Für jeden der zulässigen Mediane $c_{neu} \notin C_{i-1}$ prüfen, ob die Kosten sinken wenn $C_i = c_{i-1} \cap \{c_{neu}\}$
3. Falls ja, alle x_i mit $d(x_i, c_{neu}) < d(x_i, c_j)$ zu c_{neu} hinzu
4. Sollte ein c_j keine Punkte mehr haben, entfernen
5. Wenn $k \leq |C_i| \leq 2k$ beende Schleife
6. Wenn $|C_i| < k$: $z_{\max} = z$, sonst $z_{\min} = z$, $z = \frac{z_{\max} - z_{\min}}{2}$

gib C_i als Mediane zurück
Laufzeit $O(nk_l + nk \log k)$, $k_l = \#$Mediane in Startlösung

Primal-Dual-Algorithmus

- Basiert auf Integer Linear Programming
- Implementierung extrem komplex, sprengt den Rahmen
- Idee: Betrachte Problem als bipartiten Graphen mit Kanten zwischen Punkten und möglichen Medianen, Kantenkosten = Distanz, finde das MIN VERTEX COVER
- Laufzeit $O(m \log m(L + \log n))$, m Kanten, $L = \max \log d$
- Mediane unbekannt $\Rightarrow$ kompletter Graph: $m = \frac{n^2-n}{2}$

Beispiel

- $n_0 = 12000,\ M = 60,\ k = 3$
- $\frac{M}{k} = 20$ Punkte zu $2k = 6$ Mediane clustern
- $12000 \Rightarrow 3600 \quad (i = 1)$

- $M = 60 \Rightarrow 2k = 6$
- $3600 \Rightarrow 360 \quad (i = 2)$
- $360 \Rightarrow 36 \quad (i = 3)$

- da $36 \leq M = 60$: $36 \Rightarrow k = 3$

Komplexitätsanalyse

- Autoren postulieren: Laufzeit $\tilde{O}(n_0 k)$
- Nachrechnen:
  1. Randomized Algorithmus:
     $\frac{n_0 k}{M}$ Blöcke, je $n_1 = \frac{M}{k}$ Punkte
     1. $\sqrt{n_1 k}$ Punkte mit Primal-Dual $\Rightarrow O(n_1 k \log n_1 k)$
     2. Berechne $dist(x_i, c_j)$ für alle $i \in (1..n), j \in (1..k) \Rightarrow O(n_1 k)$
     3. $\frac{n_1}{\sqrt{n_1 k}}$ Punkte mit Primal-Dual $\Rightarrow O(\frac{n_1}{k} \log \frac{n_1}{\sqrt{n_1 k}})$

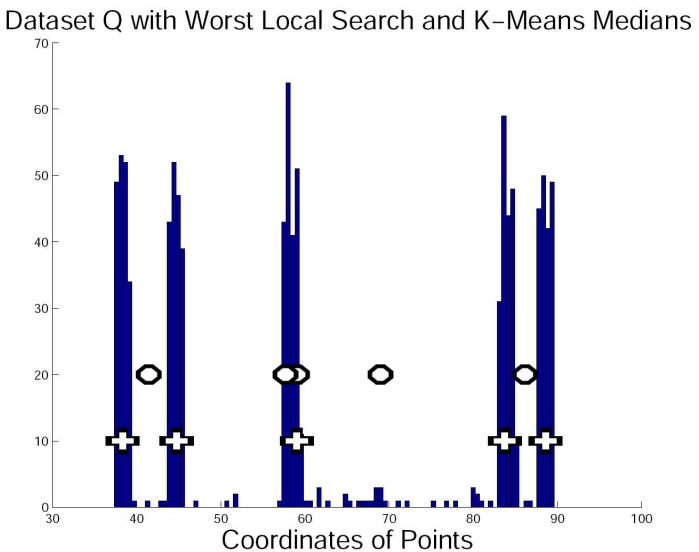
     $\Rightarrow O(n_1 k \log n_1 k)$ pro Block, $O(n_0 k \log n_0 k)$ gesamt
- Da danach die Punktmenge $\ll n$ ist fallen die folgenden Terme nicht mehr ins Gewicht
- $\Rightarrow O(n_o k \log n_0 k) \equiv \tilde{O}(n_0 k)$

Und was hat das mit Data Streams zu tun?

- Der Algorithmus arbeitet auf einer festen Grundmenge N
- Die Autoren führen lediglich an, dass die Laufzeit linear und der Speicherbedarf sublinear ist und damit tauglich für Datenströme
- Mögliche Verwendung in Datenströmen:
  - Daten puffern und die Mediane neu berechnen $\Rightarrow$ nur letzte Daten werden betrachtet
  - Mediane der alten Lösung und neue Punkte zusammen als Grundmenge nehmen, Mediane mit den Gewichten der Punkte darin, neue Punkte mit Gewicht 1
- Den Ergebnissen nach verwenden die Autoren das 2. Verfahren

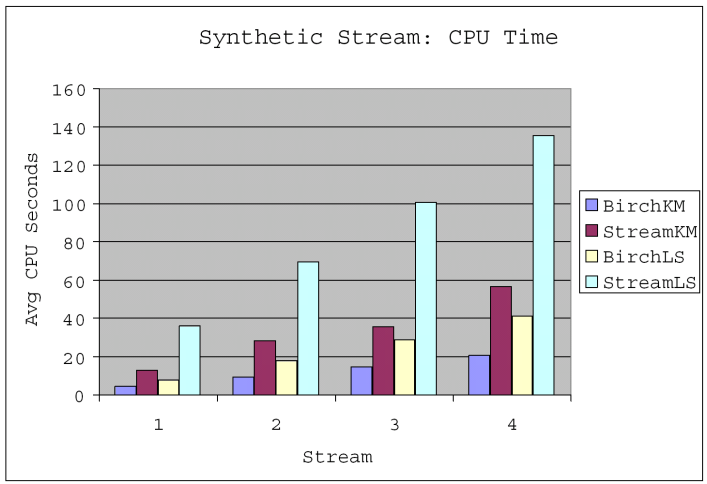
Worst Case Performance

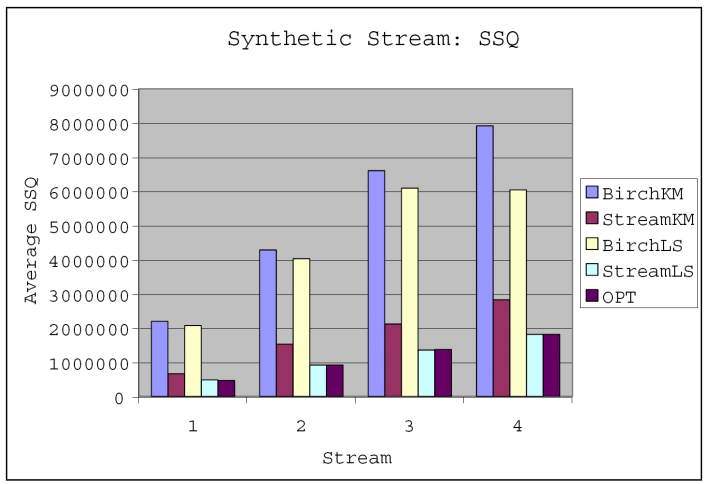
- Alle Messungen wurden 10 mal durchgeführt und gemittelt
- k-Median liefert selbst im Worst Case gute Ergebnisse im Vergleich zu k-Means (wird von Philip vorgestellt)
- Preis: 3-6-fache Laufzeit



Vergleiche syntherische Daten

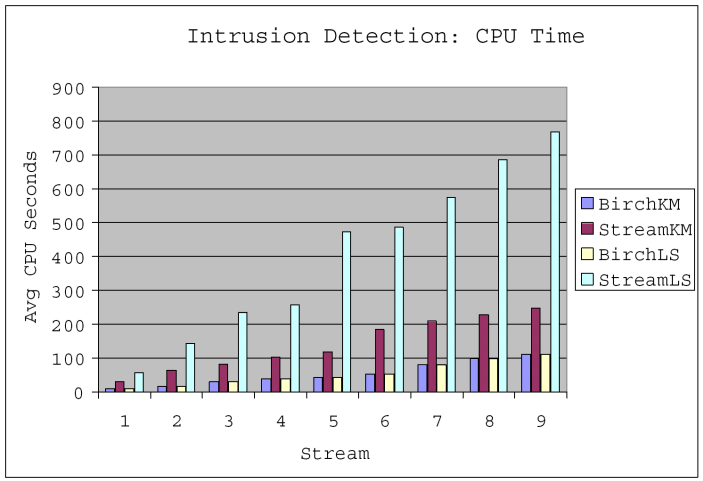
- Beispiele für LSEARCH und k-Means verwendet in BIRCH und diesem Clusteringalgorithmus
- BIRCH - deterministischer Algorithmus
- STREAMLS liefert nahezu optimale Ergebnisse
- Stochastischer Ansatz in STREAM bringt deutliche Verbesserung, auch mit k Means
- BIRCH deutlich ungenauer aber deutlich schneller

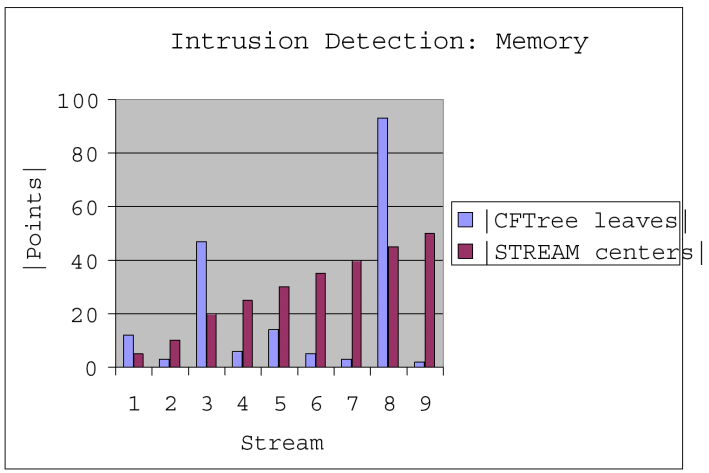




Vergleiche reale Daten

- Tradeoff Performance $\leftrightarrow$ Zeit
- Unbrauchbar für Schnelle Datenströme (Webclicks, ...)
- Gut für langsame Datenströme mit Bedarf an Präzision (IDS, ...)
- Vergleich für IDS Simulation einer Airforce Base
- 9 Blöcke TCP Rohdaten je 16MB in 2 Wochen





Quellen

- Guha, Sudipto *et al*, 2003, Clustering Data Streams: Theory and Practice
- Guha, Sudipto *et al*, 2000, Clustering Data Streams
- Kamal Jain and Vijay V. Vazirani, 1999, Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems
- Charikar, Guha, 1999, Improved Combinatorial Algorithms for the Facility Location and k-Median Problems
- ...
- Wikipedia, http://www.wikipedia.org