



Clusteranalyse und Räumliches Data Mining

Tobias Scheffer
Steffen Bickel

Cluster

- Gruppen zueinander ähnlicher Instanzen
- Text
 - ◆ Entdecken „natürlicher“ Kategorien.
- Marketing
 - ◆ Optimale Abdeckung eines Raumes von Kundenwünschen durch n Produkte.
- Astronomie
 - ◆ Clustering von Sternen
- Prozesssteuerung
 - ◆ Identifikation von Prozesszuständen

Clusteranalyse

- Hierarchische Clusterverfahren
 - ◆ Gegeben Instanzen und Ähnlichkeitsmaß, bestimme einen Baum hierarchischer Cluster, der die Ähnlichkeit der Instanzen ausdrückt.
 - ◆ Matrix-Update-Algorithmus.
- Partitionierende Clusterverfahren
 - ◆ Gegeben Instanzen, Anzahl der Cluster, angenommene parametrische Mischverteilung, finde die Parameter der Mischverteilung und damit die Clusteraufteilung der Instanzen.
 - ◆ K-Means, K-Medoids, EM.
- Skalierbare Clusterverfahren.
 - ◆ Analyse großer Datenbanken, BIRCH.
- Dichtebasierte Clusterverfahren, räumliches Data Mining.
 - ◆ Gegeben räumlich angeordnete Instanzen und Nachbarschaftsrelation, finde Bereiche hoher Dichte.
 - ◆ GDBScan.

3

Hierarchische Clustersuche

- Problemstellung
 - ◆ Finde Folge geschachtelter Cluster C^1, \dots, C^m ,
 - ◆ mit $C^i = \{C_{1}^i, \dots, C_{k}^i\}$, $C^m = D$ und alle Instanzen sind auf jeder Hierarchieebene in genau einem Cluster.
- Jedes Clusterverfahren kann rekursiv aufgerufen hierarchische Cluster erzeugen.

4

Dendrogramme

- Visualisieren Clusterhierarchien.
 - $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$
 - $\{\{1, 2\}, \{3\}, \{4\}, \{5\}\}$
 - $\{\{1, 2\}, \{3, 4\}, \{5\}\}$
 - $\{\{1, 2, 3\}, \{4, 5\}\}$
 - $\{\{1, 2, 3, 4, 5\}\}$
-
- 1 2 3 4 5
- Tiefe eines Querbalken zeigt an, auf welcher Hierarchieebene Cluster zusammengelegt werden.
 - Meist ein Maß für deren Ähnlichkeit.

5

Hierarchische Clusteranalyse

- Bottom-up (agglomerativ):
 - ◆ Beginne mit $C^1 = \{\{x_1\}, \dots, \{x_m\}\}$
 - ◆ Für i von 2 bis m
 - ★ Finde die beiden ähnlichsten Cluster in C^{i-1} .
 - ★ C^i entsteht durch zusammenfügen dieser Cluster.
- Top-Down:
 - ◆ Beginne mit $C^1 = \{\{x_1, \dots, x_m\}\}$
 - ◆ Für i von 2 bis m
 - ★ Finde heterogenstes Cluster in C^{i-1} .
 - ★ C^i entsteht durch aufspalten dieses Clusters.

6

Abstandsgraphen

- Bilde sortierte Liste der Abstände zwischen allen Instanzpaaren $\text{dist}(x_i, x_j)$, $i < j$:
 - ◆ $\langle d_1, \dots, d_{n(n-1)/2} \rangle$
- Abstandsgraph $G(k)$:
 - ◆ Knoten = Instanzen D
 - ◆ Kanten = $\{(x_i, x_j) \mid \text{dist}(x_i, x_j) < d_k\}$

7

Single-Link agglomeratives Clustering

- $C^1 = \{\{x_1\}, \dots, \{x_m\}\}$; $k=1$
- Solange C^k noch mindestens zwei Elemente hat
 - ◆ Bestimme $G(k)$
 - ◆ Wenn Anzahl Partitionen in $G(k) < |C^k|$ dann $C^{k+1} =$ Partitionen von $G(k)$
 - ◆ Inkrementiere k .
- Ergebnis ist C^1, \dots, C^k
- Beispiel

8

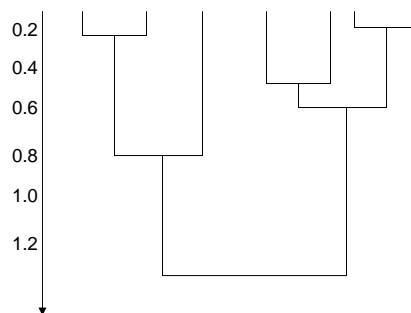
Complete-Link agglomeratives Clustering

- $C^1 = \{\{x_1\}, \dots, \{x_m\}\}; k=1$
- Solange C^k noch mindestens zwei Elemente hat
 - ◆ Bestimme $G(k)$
 - ◆ Wenn zwei Partitionen C_i^k, C_j^k eine Clique bilden, dann $C^{k+1} = \text{Cliques in } G(k)$
 - ◆ Inkrementiere k .
- Ergebnis ist C^k, \dots, C^k
- Beispiel

9

Abstands-dendrogramme

- Abstand d_k , bei dem die Cluster zusammengefügt werden
- Visualisieren Ähnlichkeit von Instanzen und Clustern



10

Matrix-Update-Algorithmus

- Implementiert single-link und complete-link agglomerative clustering.
- Zentrale Datenstruktur: Abstandsmatrix D^i .
- $D^k(i, j)$

11

Matrix-Update-Algorithmus

- $C^1 = \{\{x_1\}, \dots, \{x_m\}\}; k=1$
- $D^1(i, j) = \text{dist}(\{x_i\}, \{x_j\}) = \text{dist}(x_i, x_j)$
- Solange C^k noch mindestens zwei Elemente hat
 - ◆ $(q, s) = \text{argmin } D^k(i, j)$
 - ◆ $C^{k+1} = C^k \setminus \{C^k_q, C^k_s\} \cup (C^k_q \cup C^k_s)$
 - ◆ $D^{k+1}(i, j) = D^k(i, j)$ für $i \neq q, j \neq q$
 - ◆ $D^{k+1}(q, j) = f(D^k(q, j), D^k(s, j))$
 - ◆ $D^{k+1}(i, q) = f(D^k(i, q), D^k(i, s))$
 - ◆ Delete $D^k(i, s), D^k(s, i)$
 - ◆ Inkrementiere k .
- Ergebnis ist C^k, \dots, C^k
- Single link: $f = \text{minimum}$; complete link: $f = \text{maximum}$

12

Matrix-Update-Algorithmus

- Schnell und einfach zu implementieren.
- D^k muss in den Hauptspeicher passen (quadratischer Speicheraufwand in Anzahl der Instanzen).
- Nur für kleinere Datenmengen möglich.
- Partitionierende Clusterverfahren:
 - ◆ Keine detaillierte Betrachtung der (quadratisch vielen) Distanzen.
 - ◆ Stattdessen werden nur Distanzen zwischen Instanzen und Clustermittelpunkten betrachtet.
 - ◆ Greedy-Algorithmen.

13

Partitionierende Clusterverfahren

- Generieren Partitionierung auf einer einzelnen Hierarchieebene.
- Hierarchisches Clustern möglich durch rekursiven Aufruf.
- Hill-Climbing-Suche: Iteration der Schritte
 - ◆ Verschieben der Clustermittelpunkte
 - ◆ (Probabilistische) Zuordnung der Instanzen zu den Clustern.
- Linearer Aufwand pro Iteration

14

K-Means Algorithmus

Finde k Clustermittelpunkte. Eingabe: Instanzen, k.

1. Setze k Mittelpunkte zufällig
 2. Wiederhole
 1. Ordne jede Instanz dem nächstgelegenen Clustermittelpunkt zu.
 2. Setze die Clustermittelpunkte in den Schwerpunkt der zugeordneten Punkte.
 3. Solange sich noch Zuordnungen ändern.
- Aufwand: $n \times k \times \text{\#Iterationen}$

15

K-Means

- Extrem effizient.
- Leicht zu implementieren.
- Ergebnis hängt stark von der Initialisierung ab.
- Anzahl der Cluster muss vorher bekannt sein.
- Lokales Optimum
- Mittelwert mehrerer Objekte muss definiert sein (kein Problem bei Vektoren, Problem bei komplexeren Repräsentationen).
- K-Medoids: Statt Mittelwert zentrale Instanz.
- EM: Statistisches Modell.

16

K-Medoids

Finde k zentrale Instanzen. Eingabe: Instanzen, k .

1. Wähle k Instanzen zufällig als Medoide.
 2. Wiederhole
 1. Ordne jede Instanz dem Cluster mit nächstgelegenen Medoid zu.
 2. Für alle Cluster, für alle nicht-medoide Instanzen,
 1. Probiere aus, ob sich die Summe der Abstände der diesem Medoid zugeordneten Instanzen zum Medoid verringert, wenn die gewählte nicht-medoidale Instanz als neuer Medoid verwendet wird.
 2. Wenn Verbesserung, dann nimm neuen Medoid.
 3. Solange sich noch Zuordnungen ändern.
- Aufwand: $n^2 \times k \times \text{\#Iterationen}$

17

Expectation-Maximization-Algorithmus

- Gegeben: Daten $x_i=(y_i, z_i)$ mit sichtbarem x_i und unbekanntem z_i .
 - ◆ Hier: Clusterzugehörigkeit
- Gesucht: Modell
 - ◆ Hier: Beschreibung der Cluster in Form der Mittelpunkte.
- Dazu brauchen wir Werte der fehlenden Parameter z_i .
 - ◆ Hier: Clusterzuordnung.
- K-Means ist einfacher Spezialfall des EM-Algorithmus.

18

EM-Algorithmus (informell)

- Beginne mit zufälligem Modell (hier: zufälligen Clustermittelpunkten)
- Wiederhole
 - ◆ E-Schritt (expectation): Schätze die unbekannt Parameter auf Grundlage des Modells; bestimme $P[z_i | \text{Modell}]$.
 - ◆ M-Schritt (maximize): Finde wahrscheinlichstes Modell gegeben die geschätzten Parameter; bestimme $\text{argmax}_{\text{Modell}} P(\{y_i, z_i\} | \text{Modell})$ auf Grundlage von $P[z_i | \text{Modell}]$.

19

Probabilistische Clusterzuordnung

- Wahrscheinlichkeit dafür, dass x_i aus Cluster j stammt:
- $$E[z_{ij}] = P(\text{Modell}_j | x_i)$$
$$= \frac{P(x_i | \text{Modell}_j)P(\text{Modell}_j)}{\sum_k P(x_i | \text{Modell}_k)P(\text{Modell}_k)}$$
- $P(\theta_j)$: $P(\text{Modell}_j)$.

$$E[z_{ij}] = \frac{P(x_i | \mu_j, \sigma_j)P(\theta_j)}{\sum_k P(x_i | \mu_k, \sigma_k)P(\theta_k)}$$

20

EM-Algorithmus für univariate Normalverteilungen

- Beginne mit zufälligen Werten für alle θ_j .
- Wiederhole
 - ◆ E-Schritt: Für alle Datenpunkte i :

$$E[z_{ij}] = \frac{P(x_i | \mu_j, \sigma_j) P(\theta_j)}{\sum_k P(x_i | \mu_k, \sigma_k) P(\theta_k)}$$

- ◆ M-Schritt: Schätze alle μ_j, σ_j :

$$\mu_j = \frac{\sum_{i=1}^m E[z_{ij}] x_i}{\sum_{i=1}^m E[z_{ij}]} ; \sigma_j = \sqrt{\frac{\sum_{i=1}^m E[z_{ij}] (x_i - \mu)^2}{\sum_{i=1}^m E[z_{ij}]}}$$

21

Partitionierendes Clustern

- Aufwand:
 - ◆ In jeder EM-Iteration mindestens eine Iteration über alle Daten, um z_{ij} und Parameter zu bestimmen.
 - ◆ Anzahl der Iterationen nicht zu überblicken.
 - ◆ Zu viel für große Datenbanken.
- Idee von BIRCH
 - ◆ einmal über die Beispiele iterieren,
 - ◆ Dabei inkrementell Cluster aufbauen,
 - ◆ Nur Parameter der Cluster abspeichern, nicht die Beispiele selbst.

22

BIRCH: Cluster Features

- Parameter, die ein Cluster beschreiben: (N, LS, SS)
 - ◆ N: Anzahl der Punkte im Cluster.
 - ◆ Vektor LS: Summe der Punkte im Cluster: $LS = \sum_i X_i$
 - ◆ Vektor SS: Quadratische Summe der Punkte im Cluster: $SS = \sum_i X_i^2$
- Aus den Cluster Features lassen sich
 - ◆ Mittelwert: $X_0 = \frac{1}{N} \sum_i X_i = \frac{1}{N} LS$
 - ◆ Radius (Varianz):

$$R = \sqrt{\frac{1}{N} \sum_i (X_i - X_0)^2} = ?$$
- bestimmen.

23

BIRCH: Cluster Features und CF Tree

- Additivität:
 - ◆ Wenn zwei Cluster durch (N_1, LS_1, SS_1) und (N_2, LS_2, SS_2) beschrieben werden,
 - ◆ Dann hat das zusammengelegte Cluster die Eigenschaften $(N_1+N_2, LS_1+LS_2, SS_1+SS_2)$.
- „Cluster Feature Tree“, höhenbalancierter Baum mit Parametern
 - ◆ Verzweigungsfaktor B und
 - ◆ Schwellenwert (max Radius der Instanzen im Blattknoten) T.
 - ◆ Knoten der vorletzten Ebene enthalten jeweils höchstens L Blattknoten.

24

Einfügen von Instanz x in den CF Tree

- Identifiziere den Knoten n der vorletzten Ebene:
 - ◆ Nimm von der Wurzel aus jeweils die Verzweigung zum Knoten wählen, die der einzufügenden Instanz näher liegt.
- Identifiziere das Blatt, das der neuen Instanz am nächsten liegt.
 - ◆ Wenn neue Instanz hinzugefügt werden kann ohne dass dabei der Radius T überschritten wird, hat das neue Blatt die Cluster Features $(N+1, LS+x, SS+x^2)$.
 - ◆ Sonst: Wenn dem Knoten n noch ein Blatt hinzugefügt werden kann, bilde ein neues Blatt für x.
 - ◆ Sonst: Spalte n in zwei Knoten auf. Finde das Paar verschiedenster Blätter und teile sie auf die beiden neuen Knoten auf. Teile die anderen Blätter den beiden Knoten nach Ähnlichkeit zu.
 - ◆ Wenn nötig, spalte Elternknoten von n auf usw. bis zum Wurzelknoten.

25

BIRCH

- „Verfeinerung der Splits“. Aufspaltung der Knoten hängt nur von Anzahl der Kinder ab, nicht von Clusterqualität.
- Verfeinerungsschritt:
 - ◆ Aufwärtspropagation der Aufspaltung endet bei Knoten n.
 - ◆ Wähle die ähnlichsten Kinder von n. Lege die beiden Kinder zusammen.
 - ◆ Wenn dadurch die Kapazität B des zusammgelegten Kindknoten überschritten wird, werden die Knoten wieder aufgespalten. Auch dann kann das Ergebnis eine bessere Aufteilung sein.

26

BIRCH

1. Iteration über alle Datenbankeinträge:
 1. Füge Eintrag in CF-Baum ein.
2. Starte jetzt einen Clusteralgorithmus, der nur noch auf den Blättern des CF-Baumes arbeitet.

27

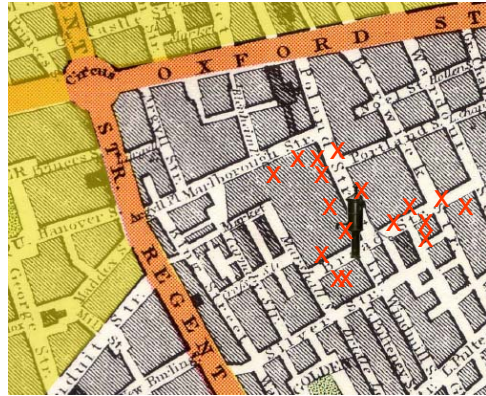
Räumliches Data Mining

- Clusteranalyse räumlicher Daten
- Entdecken von Strukturen auf Karten
- Analyse von
 - ◆ Zensus-Daten
 - ◆ Grundstückspreisen
 - ◆ Ausbreitung von Krankheiten
 - ◆ Klimadaten

28

Cholera-Mining

- Identifizieren zusammenhängender Gebiete hoher Punktdichte.



29

Räumliche Daten

- Punktförmige Daten: x-, y-, ggz. z-Koordinaten.
- Flächen: Kreise, Polygone, ...
- Unterschied zu anderen Daten:
 - ◆ Nachbarschaftsrelation
 - ◆ Distanzrelation
- Clusteranalyse räumlicher Daten
 - ◆ Häufig keine konvexen Cluster.

30

Dichte-basierte Clusteranalyse

- Idee
 - ◆ Hohe Dichte an Datenpunkten innerhalb der Cluster
 - ◆ Niedrige Dichte ausserhalb.
 - ◆ Form der Cluster unerheblich
- Erfordert die Definition der „Dichte“ in der Nachbarschaft von Punkten.
- Algorithmus GDBSCAN.

31

Nachbarschafts-Prädikat

- Z.B.
 - ◆ $(x,y) \in \text{Npred} \Leftrightarrow \{ \text{dist}(x, y) < \varepsilon \}$ (für Punkte)
 - ◆ $(x,y) \in \text{Npred} \Leftrightarrow \{ x \text{ schneidet } y \}$ (für Flächen)
- $\text{Npred}(x) = \{ y \mid \text{Npred}(x, y) \}$
- Nachbarschaft muss reflexiv und symmetrisch sein:
 - ◆ $\forall x: \text{Npred}(x, x)$
 - ◆ $\text{Npred}(x, y) \Rightarrow \text{Npred}(y, x)$

32

Mindest-Dichte

- **MinWeight (S):** Hat Menge S eine hohe Dichte?
 - ◆ $\text{MinWeight (S)} \leftrightarrow \text{wCard (S)} \geq \text{MinCard}$
- **wCard (S);** mögliche Definitionen:
 - ◆ Anzahl der Punkte in S.
 - ◆ Gewichtete Anzahl (z.B. mit Einkommen gewichtet).
 - ◆ Gewichtet mit Fläche von S.

33

Dichte-verbundene Mengen

- Objekt p ist von Objekt q *direkt erreichbar*, wenn
 - ◆ $N_{\text{pred}}(p, q)$ (p und q sind benachbart)
 - ◆ $\text{MinWeight}(N_{\text{pred}}(q))$ (q liegt in Nachbarschaft mit hoher Dichte)
- Keine symmetrische Relation (Beispiel?)
- Erreichbar: transitiver Abschluss von direkt erreichbar.

34

Dichte-verbundene Mengen

- P ist dichte-verbunden mit q, wenn
 - ◆ Es ein Objekt o gibt, so dass
 - ◆ P erreichbar von o und
 - ◆ Q erreichbar von o ist.

- Eine Menge C in einer Menge D ($C \subset D$) ist dichte-verbunden, wenn
 - ◆ $\forall p, q \in C$: p und q sind dichte-verbunden
 - ◆ $\forall p, q \in D$: Wenn q von p erreichbar ist und $p \in C$, dann gilt auch $q \in C$.

35

Dichte-verbundene Mengen

- Wenn $\text{MinWeight}(N_{\text{pred}}(p))$ gilt, dann ist $\{q \in D \mid q \text{ ist erreichbar von } p\}$ eine dichte-verbundene Menge.

- Wenn C eine dichte-verbundene Menge in D ist und $p \in C$ so dass $\text{MinWeight}(N_{\text{pred}}(p))$, dann gilt $C = \{q \in D \mid q \text{ ist erreichbar von } p\}$.

- Wir können dichte-verbundene Mengen dadurch konstruieren, dass wir von einem Kernobjekt zu allen dichte-verbundenen Objekten gehen.

36

Dichte-verbundene Cluster

- Eine dichte-verbundene Clustering $CL(S)$ einer Objektmenge S ist die Menge aller dichte-verbundenen Mengen in S .
- Es kann Punkte ohne Clusterzugehörigkeit geben.
- Cluster können sich an Randpunkten überlappen wenn $MinCard$ mindestens 4 ist.
- Es gibt genau eine Clustering.
- Beispiel?

37

GDBSCAN

- GDBSCAN ($S, Npred, MinCard$)
 1. ClusterID = 1
 2. For all o in S
 1. If ClusterID (o) = { } then
 1. If Expand ($S, o, ClusterID, Npred, MinWeight$) then increment (ClusterID).

38

GDBSCAN

```
ExpandCluster(SetOfObjects, Object, ClId, NPred, MinCard, wCard): Boolean;
IF wCard({Object}) ≤ 0 THEN // point not in selection
  SetOfPoints.changeClId(Object, UNCLASSIFIED);
  RETURN False;
END IF
seeds := SetOfObjects.neighborhood(Object, NPred);
IF wCard(seeds) < MinCard THEN // no core point
  SetOfObjects.changeClId(Object, NOISE);
  RETURN False;
END IF
// still here? Object is a core object
SetOfObjects.changeClIds(seeds, ClId);
seeds.delete(Object);
WHILE seeds ≠ Empty DO
  currentObject := seeds.first();
  result := SetOfObjects.neighborhood(currentObject, NPred);
  IF wCard(result) ≥ MinCard THEN
    FOR i FROM 1 TO result.size DO
      P := result.get(i);
      IF wCard({P}) > 0 AND P.ClId IN {UNCLASSIFIED, NOISE} THEN
        IF P.ClId = UNCLASSIFIED THEN
          seeds.append(P);
        END IF;
        SetOfObjects.changeClId(P, ClId);
      END IF; // wCard > 0 and UNCLASSIFIED or NOISE
    END FOR;
  END IF; // wCard ≥ MinCard
  seeds.delete(currentObject);
END WHILE; // seeds ≠ Empty
RETURN True;
END; // ExpandCluster
```

39

GDBSCAN

- Komplexität:
 - ◆ $O(n * \text{Aufwand für Nachbarschaftsanfrage})$
 - ◆ Nachbarschaftsanfrage in $\log n$ in SDBS (Spatial DBS)
 - ◆ $O(n \log n)$
- Anwendungen
 - ◆ Clustersuche auf Satellitenbildern.
 - ◆ Astronomie: Entdecken von Strahlungsquellen auf Radiointensitätsbildern.
 - ◆ Geographie: Einzugsgebiete von Städten.

40