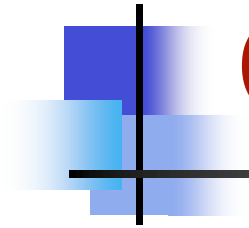
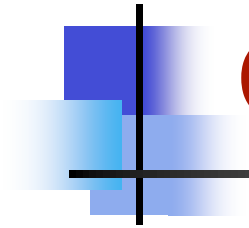


Clustering

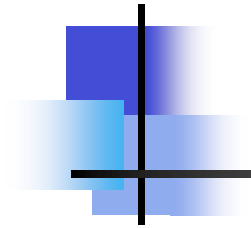
Hauptseminar Machine Learning
WS 2003/2004

Referent: Can Önder
Betreuer: Martin Wagner



Gliederung

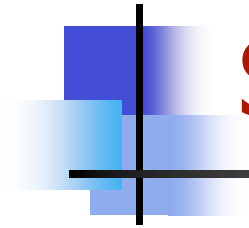
- Partitionierendes Clustering
- Hierarchisches Clustering
- Wahrscheinlichkeitsbasiertes Clustering



Definition

- Ein Cluster ist eine Menge von Objekten, die untereinander eine hohe und zu anderen Objekten außerhalb des Clusters eine möglichst geringe Ähnlichkeit aufweisen

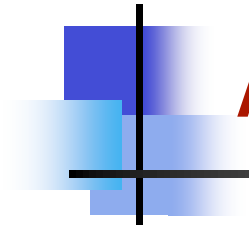
- Oft: Finden „natürlicher Gruppen“ in den Daten



Segmentierung vs. Clusteranalyse

- Segmentierung:
  Zweckmäßige Unterteilung der Daten

- Clusteranalyse:
  - Finden von Strukturen in den Daten (natürliche Klassen)
    z.B. Medizin, Biologie, Marktsegmentierung, Kaufverhalten
  - Aufstellen von Statistiken



Abstandsfunktionen

- Wie misst man die „Ähnlichkeit“ von Objekten?
- Distanzfunktionen unterschiedlich je nach Anwendung (viele Ansätze und Weiterentwicklungen)
- Quantitative (numerische) vs. nicht-quantitative (nominale) Daten-Attribute

Wahl der Abstandsfunktion ist entscheidend für das resultierende Ergebnis (wichtiger als der Algorithmus)!

Abstandsfunktionen

- Euklidischer Abstand:

$$dist(x,y) = \sqrt{(x_1 - y_1)? + \ldots + (x_n - y_n)?} \quad = \sqrt{\sum_{i=1}^{n} (xi - yi)?} \quad = \left\| x_i - y_i \right\|?$$

- Abstandsfunktion basierend auf Attributen:

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} d_j \cdot (x_{ij}, x_{i'j}); \qquad d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})?$$

Gewichtet: $$D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \bullet d_j(x_{ij}, x_{i'j}); \qquad \sum_{j=1}^{p} w_j = 1$$

- Proximity Matrices:

$$\begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1N} \\ d_{21} & \ddots & & \\ \vdots & & \ddots & \\ d_{N1} & \cdots & & d_{NN} \end{pmatrix}$$

Abstandsfunktionen

- Nearest-neighbor:

$$D(C_i, C_j) = \min_{x,y} \{d(x,y) \mid x \in C_i, y \in C_j\}$$

- Furthest neighbor:

$$D(C_i, C_j) = \max_{x,y} \{d(x,y) \mid x \in C_i, y \in C_j\}$$

- Beispiel für nominale Attribute:

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

- Weitere:

  Hamming-Abstand, Zentrums-Messungen (als Cluster-Abstand) und viele mehr...

Abstandsfunktionen

3 Bedingungen für Distanzfunktionen:

1. $\forall x, y \in D : dist(x, y) \in R^{+}$
2. $dist(x, y) = 0 \Leftrightarrow x = y$
3. $dist(x, y) = dist(y, x)$

Partitionierendes Clustering

- Optimale Einteilung der Daten in eine vorgegebene Anzahl von Clustern

- Datenmenge $D = \{x_1, x_2, \ldots, x_n\}$ in k disjunkte Partitionen einteilen
- → Clustermenge C mit k Clustern: $C = \{C_1, C_2, \ldots, C_k\}$
- Jeder Punkt x_i gehört gehört in ein eindeutiges Cluster C_k
- Jedes Cluster besteht aus mindestens einem Objekt
- Jedes Objekt ist in höchstens einem Cluster enthalten

Bewertungsfunktion

- Soll die Qualität eines Clusterings ausdrücken
- Ist dazu da, eine optimale (bzw. „gute“) Partitionierung der Daten zu finden / zu suchen
- Minimierung bzw. Maximierung der Bewertungsfunktion zum Finden einer guten Lösung
- Leider oft nur lokale Extrema!
- Keine eindeutige Lösung zum Optimieren einer Bewertungsfunktion → systematische Suche nach möglichen Clustern im Datenraum

→ Viele bekannte Ansätze

Bewertungsfunktion

Partitionierendes Clustering:

Ziele:

- Cluster sollen kompakt sein
- Cluster sollen so weit wie möglich von einem anderen entfernt sein

- Within cluster variation wc(C)
- between cluster variation bc(C)

Bewertungsfunktion

Within cluster variation $wc(C)$:

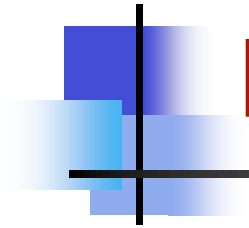
$$r_k = \frac{1}{n_k} \sum_{x \in C_k} x$$

Cluster-Zentrum

$$wc(C) = \sum_{k=1}^{K} wc(C_k) = \sum_{k=1}^{K} \sum_{x_i \in C_k} d(x, r_k)?$$

Between cluster variation $bc(C)$:

$$bc(C) = \sum_{1 \le j < k \le K} d(r_j, r_k)?, \quad j \neq k$$



Bewertungsfunktion

Bewertungsfunktion:

- Gesamtqualität des Clusters
- Monotone Kombination von wc(C) und bc(C)

- bc(C) / wc(C)

Bewertungsfunktion

Alternative Ansätze für wc(C):

- „minimum distance criterion“:

$$wc(C) = \max_{i} \min_{y_i \in C_k} \{d(x_i, y_i) \mid x_i, y_i \in C_k, x \neq y\}$$

- Kovarianz-Matrix:

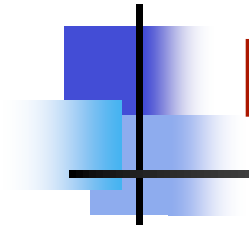
$$W_k = \sum_{x \in C_k} (x - r_k) \bullet (x - r_k)^T \Rightarrow wc(C) = \sum_k tr(W_k)$$

- oder:

$$B = \sum_{k=1}^{n} n_k \bullet (r_k - \bar{\mu})(r_k - \bar{\mu})^T$$

Als Bewertungsfunktionen verwende Kombinationen von B und W:

$$tr(W), \det(W), tr(BW^{-1})$$



Part. Clustering

Ansatz:

alle Cluster-Einteilungen durchprobieren und für die beste entscheiden → ineffizient, da exponentielle Laufzeit

Weitverbreitete Vorgehensweise: Iterative Verfahren

- Starte mit initialem Clustering
- Ordne die Objekte iterativ immer wieder neu zu, sodass die Bewertungsfunktion optimal ist...
- ...bis das Clustering optimal ist

→ K-means-Algorithmus!!!

K-means-Algorithmus

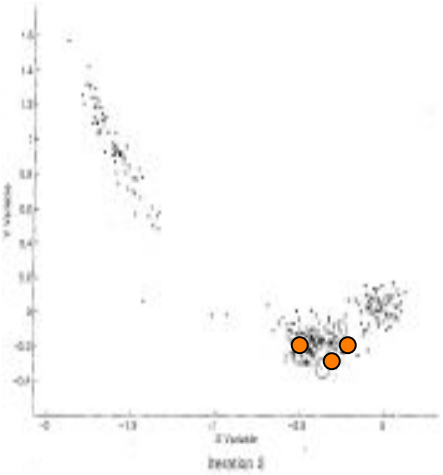
1. K = Anzahl Cluster – im Voraus festgelegt
2. Wähle zufällig k Cluster-Zentren („representatives“)
3. Weise jeden Punkt dem ihm naheliegendsten Zentrum zu (euklidische Abstandsfunktion)
4. Berechne die neuen Zentren
5. Wiederhole iterativ bei 3.

Ende erreicht, wenn die Zentren stabil sind

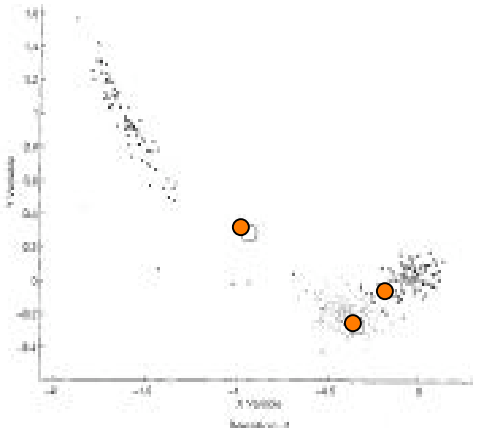
K-means-Algorithmus

- Beispiel

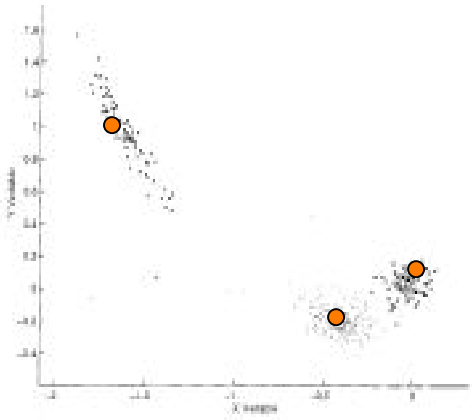
1)



2)



3)



K-means-Algorithmus

FOR k=1,...,K sei r_k ein zufälliger Punkt aus D;

WHILE es ergeben sich Änderungen im Cluster C_k DO

Cluster bilden:

FOR k=1,...,K DO

$$C_k = \{x \in D \mid d(r_k, x) \leq d(r_j, x) \forall j = 1,...,K, j \neq k\};$$

END;

Neue Cluster-Zentren bilden:

FOR k=1,...,K DO

r_k= Mittel der Punkte im Cluster C_k;

END;

END;

Pro/Contra: K-means

- kompakte, runde Cluster
- Einfach, intuitiv
- Liefert mindestens ein lokales Extremum
- Komplexität O(KnI)

- Keine Aussage über globales Maximum – besseres Clustering möglich?
- Qualität des Clusterings stark abhängig von K (muss vorher feststehen)
  (schlechte Wahl → gutes Cluster kann übersehen werden – Beispiel mit Rechteck)
- Nur für numerische Attribute
- Probleme mit „Ausreißern"

Algorithmus mehrmals laufen lassen mit unterschiedlichen Startwerten
→ bessere Chance auf globales Extremum!

Erweiterungen

Verschiedene Variationen von k-means-Algorithmus bekannt

Beispiel: K-Menoids-Clustering:

- Anwendbar auf beliebige Attribute
- Weniger Einfluss von „outliers“
- Repräsentation der Cluster nicht durch Zentren sondern geschätzte „repräsentative Beispiele“ (Menoide)
- Clusteraufteilung nach dem „nearest-neighbor“-Prinzip

- Nachteil: höherer Rechenaufwand

Hierarchisches Clustering

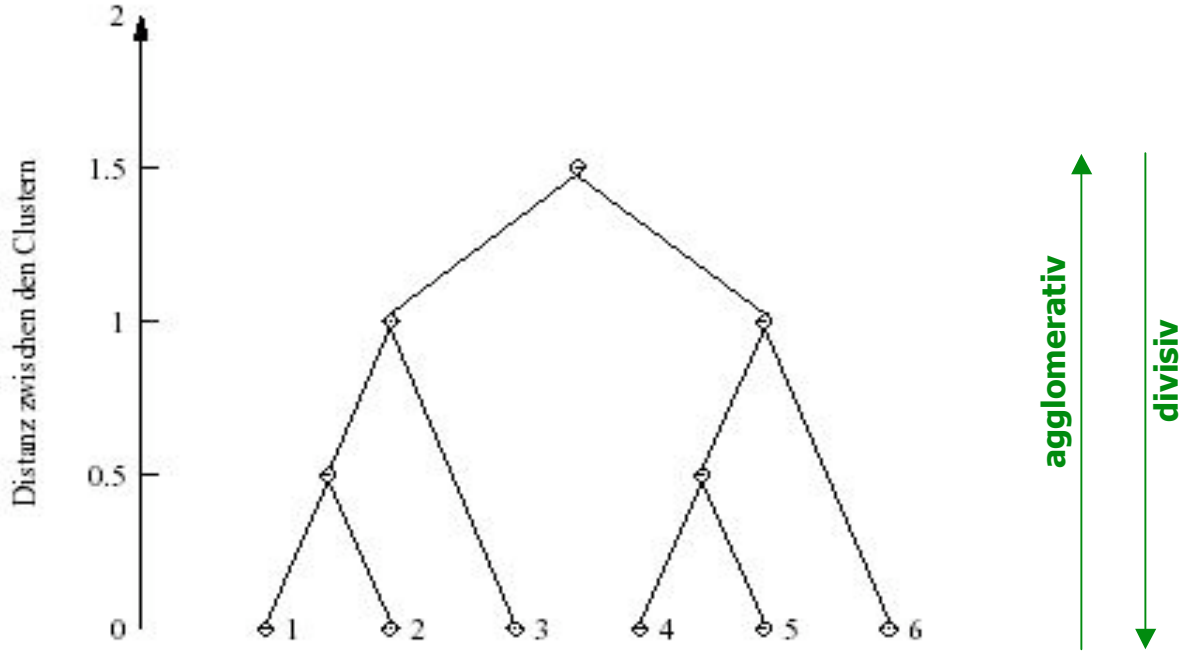
- Rekursive Unterteilung
- Graphisch gut darstellbar → Dendogramm
- Baumartige Struktur der entstehenden Cluster
  - Wurzel: gesamtes Daten-Set
  - Blätter: einzelne Objekte
  - Knoten: Gruppen / Cluster
- Sehr tiefreichende Unterteilung

2 verschiedene Ansätze:

- Agglomeratives Clustering
- Divisives Clustering

Hierarchisches Clustering

Dendogramm (Beispiel)



Hierarchisches Clustering

- Anzahl der Cluster wird erst zur Laufzeit bestimmt
- Unterschiedlichkeit der Cluster steigt mit höherem Level
- Höhe eines Knotens: Maß für die interne „Variation" seiner Kinderknoten
- Höher liegende Gruppen: natürliche Gruppen (?)

- Dendogramme sind keine 1:1- Abbildung der Daten:
  - Untersch. Methoden → untersch. Dendogramme
  - Entstehung der Baumstruktur durch Algorithmus erzwungen, unabhängig von der tatsächlichen Struktur der Daten
  - Interpretation eher als Relation der N(N-1)/2 paarweisen Objekt-Unterschiedlichkeiten

Agglomeratives Clustering

- „Bottom-up"-Methode
- Basiert auf Abstandsmessung zwischen Clustern / Gruppen
- Zu Beginn jedes Objekt in eigenem Cluster
- Immer die 2 ähnlichsten Cluster werden gemerged und gelangen ein Level höher → ein Cluster weniger
- Beim Mergen evtl. Umstrukturierung des Baumes notwendig
- Entscheidung, welche Objekte gemerged werden, mittels „Category Utility" (Berechnungsformel)
- Ende erreicht, wenn nur noch ein Cluster ganz oben vorhanden

Agglomeratives Clustering

Algorithmus:

n Datenpunkte $D = \{x_1, \ldots, x_n\}$

Abstandsfunktion $D(C_i, C_j)$ *für 2 Cluster* C_i *und* C_j

FOR i=1,...,n sei $C_i = \{x_i\}$;

WHILE es gibt noch mehr als ein Cluster DO

Seien C_i und C_j die Cluster mit geringstem Abstand $D(C_k, C_h)$ von allen Cluster-Paaren;

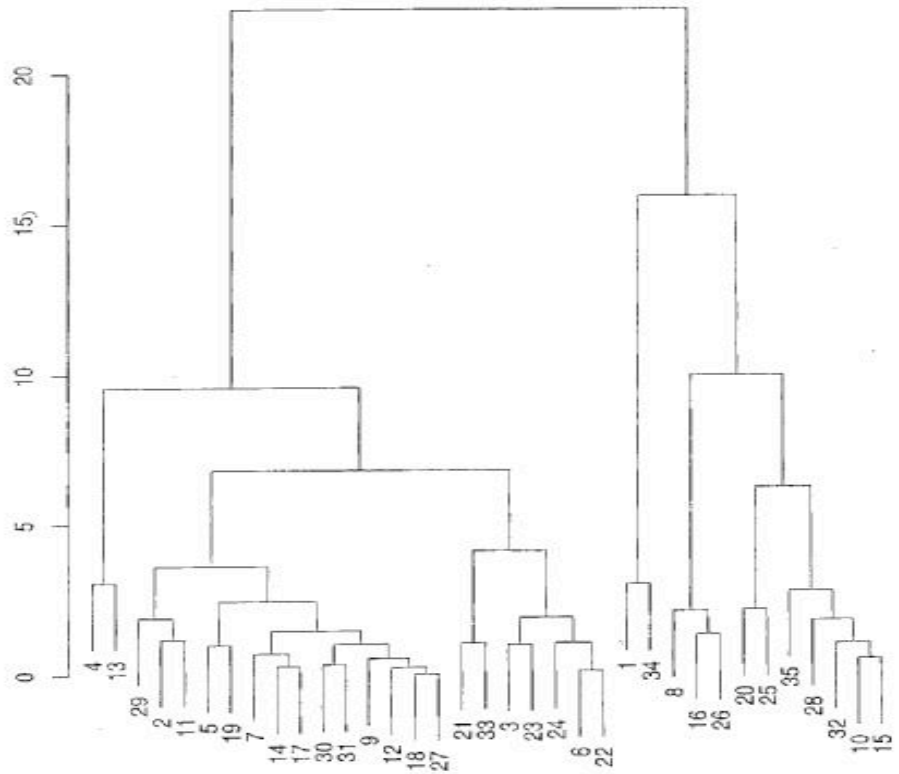
$C_i = C_i \cup C_j$;

Entferne Cluster C_j;

END;

Agglomeratives Clustering

- Dendogramm



Abstandsfunktion

- Nearest neighbor (single link) - Methode:

$$D(C_i, C_j) = \min_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

- Nachteile:
  - Verletzung der Kompaktheit der Cluster
  - Hoher Einfluss außenliegender Punkte

- Furthest neighbor (complete link) – Methode:

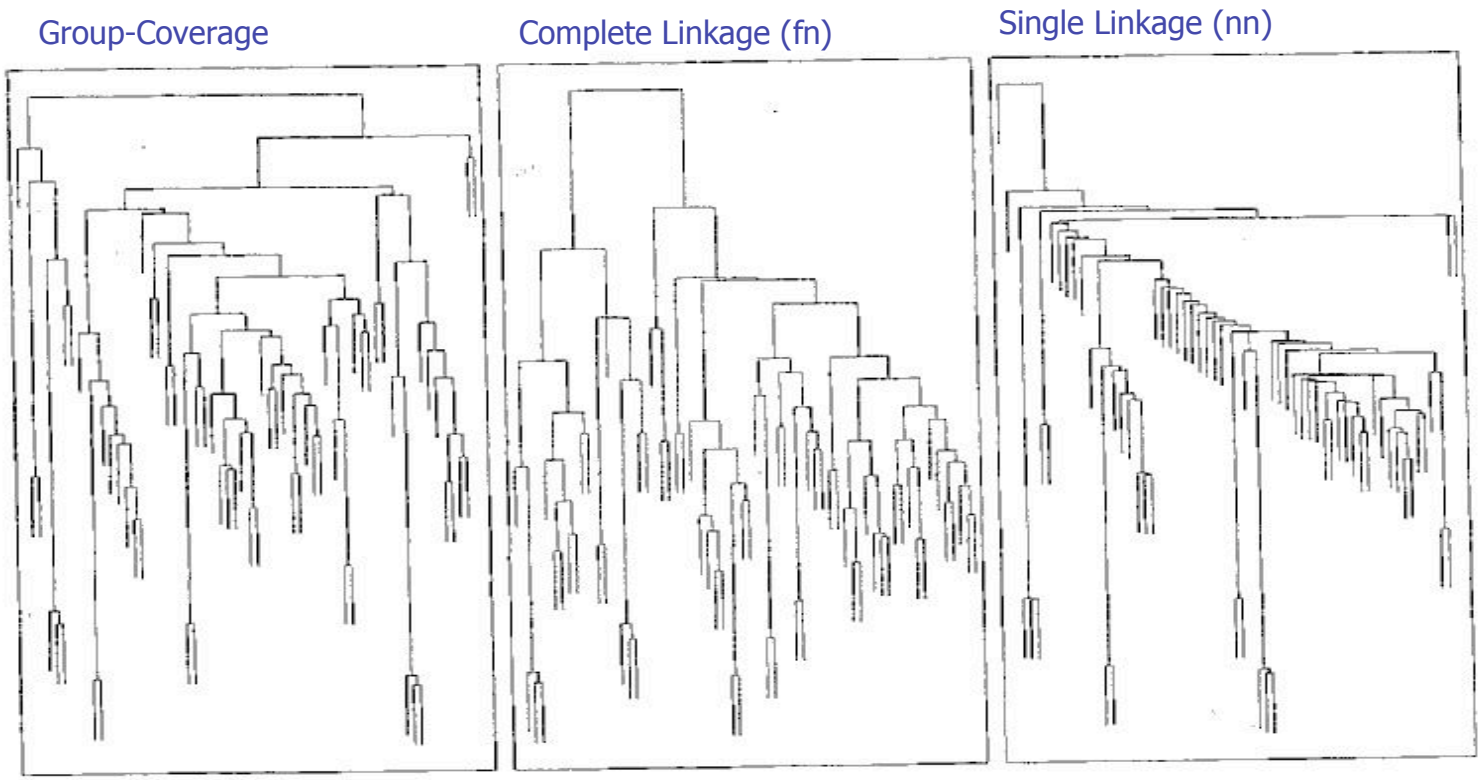
$$D(C_i, C_j) = \max_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

- Nachteile:
  - Objekte im selben Cluster können näher zu Objekten in anderen Clustern sein

→ Kompromiss: Group-Coverage (Durchschnitt)

Abstandsfunktion

- Vergleich



Category Utility

- Berechnet den Wert der Wahrscheinlichkeit eines Attributes, dass dieses Attribut in ein bestimmtes Cluster gehört
- Qualitätsmessung für eine Unterteilung von Objekten / Instanzen in Cluster

- Berechnung:

$$CU(C_1,\ldots,C_k) = \frac{\sum_l \Pr[C_l] \sum_i \sum_j (\Pr[a_i = v_{ij} \mid C_l]? - \Pr[a_i = v_{ij}]?)}{k}$$

Category Utility

- Erweiterung auf numerische Attribute
- Nehme Normalverteilung für die Attribute an

$$f(a) = \frac{1}{\sqrt{2\pi\sigma}} \bullet e^{\frac{(a-\mu)?}{2\sigma?}}$$

Wahrscheinlichkeitsdichtefunktion für ein Attribut a

$$\sum_j \Pr[a_i = v_{ij}]? \Leftrightarrow \int f(a_i)?da_i = \frac{1}{2\sqrt{\pi\sigma_i}}$$

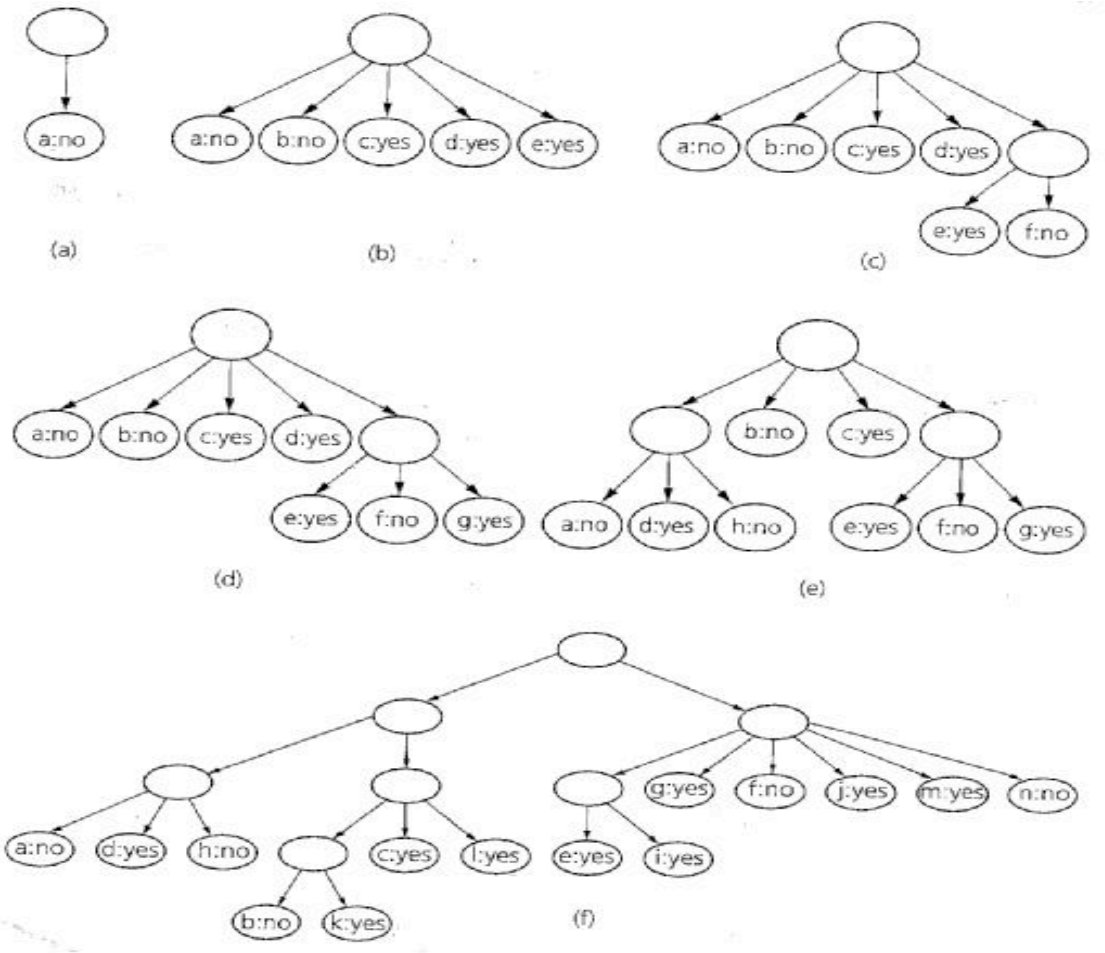
Summation der Quadrate der Attributwahrsch.

$$\Rightarrow CU(C_1,\ldots,C_k) = \frac{1}{k}\sum_l \Pr[C_l]\frac{1}{2\sqrt{\pi}}\sum_i \left(\frac{1}{\sigma_{il}} - \frac{1}{\sigma_i}\right)$$

Category Utility

Category Utility

- Beispiel:



Agglomeratives Clustering

- Komplexität (Zeit und Speicher): O(n_),
  da alle paarweisen Abstände von Clustern überprüft (und gespeichert vorliegen) müssen

- Für großes n:
  - Hoher Rechenaufwand
  - Baumstruktur wird sehr unübersichtlich

Divisives Clustering

- „Top-down"-Methode

- Algorithmus:
  - Starte mit oberstem Cluster – enthält alle Daten
  - Suche nach dem Objekt mit größtem Abstand zu allen anderen Objekten und bilde mit ihm ein neues, separates Cluster
  - Berechne für jedes Objekt im 1. Cluster:

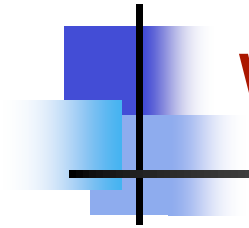
    (Durchschnittlicher Abstand zu allen Objekten im neuen Cluster) – (Durchschnittlicher Abstand zu allen verbleibenden Objekten im 1. Cluster)
  - Objekt mit größtem Ergebniswert ins neue Cluster
  - Fahre fort bis Differenz negativ wird → neues Cluster fertig!
  - Rekursives „Splitting" auf beide Unter-Cluster anwenden, bis jedes Objekt in eigenem Cluster ist

Divisives Clustering

- Weniger populär und weniger weit verbreitet als agglomeratives Clustering
- Mehr Rechenaufwand als bei agglomerativen Methoden

- großer Vorteil hierarchisches Clustering:

  K muss nicht feststehen! (Hauptproblem der anderen Verfahren: die Wahl von k)



Wahrscheinlichkeitsbasiertes Clustering

- Annahme: Objekte gehören mit einer bestimmten Wahrscheinlichkeit zu einem Cluster
- Grundlage: Mixture Models
- Hier: Mixture ist ein Satz von Wahrsch., die k Cluster darstellen
- Jedes Cluster hat seine eigene Verteilung; jede Instanz gehört zu genau einem Cluster
- Häufige Annahme: Normalverteilung; Instanzen unabhängig

Wahrscheinlichkeitsbasiertes Clustering

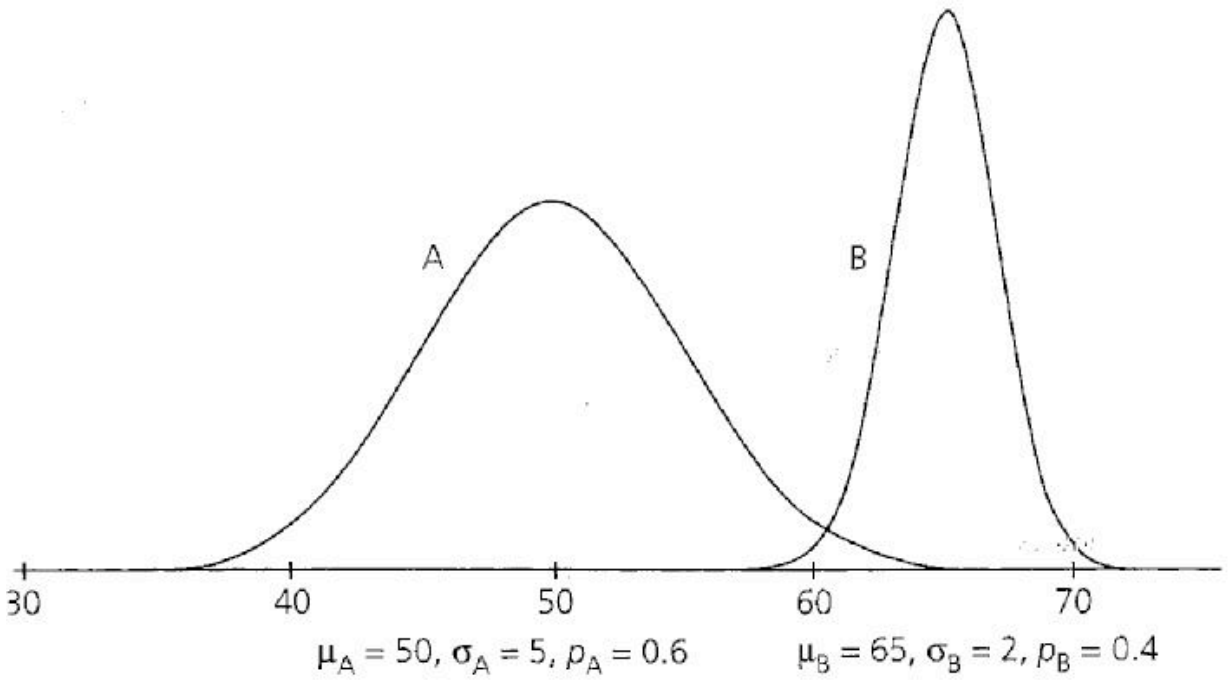
- Beispiel:
- 2 Cluster A, B, Normalverteilung liegt zugrunde mit unbekannten Parametern $\mu_A, \mu_B, \sigma_A, \sigma_B, p_A$
- Annahme: man weiß, aus welchen Clustern die Instanzen stammen → Parameter berechnen sich wie folgt:

$$\mu = \frac{x_1 + x_2 + \ldots + x_n}{n}, \qquad \sigma? = \frac{(x_1 - \mu)? + (x_2 - \mu)? + \ldots + (x_n - \mu)?}{n - 1}$$

- p_A berechnet sich aus dem Verhältnis der aus Cluster A stammenden Punkte und der Anzahl aller Punkte
- $p_B = 1 - p_A$

Wahrscheinlichkeitsbasiertes Clustering

Beispiel: Mixture Model für k=2 Cluster A, B



Wahrscheinlichkeitsbasiertes Clustering

- Kennt man die Parameter, aber nicht die Cluster-Herkunft:

$$\Pr[A \mid x] = \frac{\Pr[x \mid A] \bullet \Pr[A]}{\Pr[x]} = \frac{f(x; \mu_A; \sigma_A) \bullet p_A}{\Pr[x]}$$

mit $$f(x; \mu; \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \bullet e^{\frac{(x-\mu)?}{2\sigma?}}$$ (Normalverteilung)

Wahrscheinlichkeitsbasiertes Clustering

- Problem:

  Normalerweise sind weder die Verteilung noch deren Parameter bekannt

→ Schätzen mit EM-Algorithmus:

1. Starte mit geschätzten 5 Parametern
2. Verwende die Parameter zum Errechnen der Cluster-Wahrsch. (E)
3. Verwende diese Wahrsch., um erneut die Parameter zu schätzen (M)
4. Zurück zu 2. usw.

Wahrscheinlichkeitsbasiertes Clustering

- Man erhält die Cluster-Wahrscheinlichkeiten, nicht die Cluster selbst; die Wahrsch. verhalten sich aber wie Gewichte
- Sei ω_i die Wahrsch., dass Instanz i zum Cluster A gehört:
- → $\mu_A = \frac{\omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n}{\omega_1 + \omega_2 + \ldots + \omega_n}$ und

$$\sigma?_A = \frac{\omega_1 (x_1 - \mu)? + \omega_2 (x_2 - \mu)? + \ldots + \omega_n (x_n - \mu)?}{\omega_1 + \omega_2 + \ldots + \omega_n}$$

- → berechne wieder die Wahrscheinlichkeiten:

$$\Pr[A \mid x] = \frac{\Pr[x \mid A] \bullet \Pr[A]}{\Pr[x]} = \frac{f(x; \mu_A; \sigma_A) p_A}{\Pr[x]}$$

Wahrscheinlichkeitsbasiertes Clustering

- EM-Algorithmus konvergiert gegen Maximum, ohne es zu erreichen (Gegensatz zu k-means)
- Abbruch nach „genügend vielen" Iterationsschritten
- Maß dafür, wie „gut" das Clustering ist: Maximum Likelihood
- Nimmt mit jeder Iteration einen besseren Wert an

- Aber:
  - Auch EM-Algorithmus liefert nur lokales Extremum
  - Mehrere Durchläufe → Entscheidung für bestes Ergebnis

Wahrscheinlichkeitsbasiertes Clustering

- Erweiterung des „Mixture Models“ auf beliebig viele Cluster
- Selbe Basis-Methode
- Wahrscheinlichkeiten jedes Attributes werden multipliziert
- → Gesamtwahrscheinlichkeit für eine Instanz
- Korrelierte Attribute → Covarianz-Matrix
- Attribute mit v möglichen Werten als Vektor mit v Einträgen darstellen (Einträge: Wahrscheinlichkeiten)

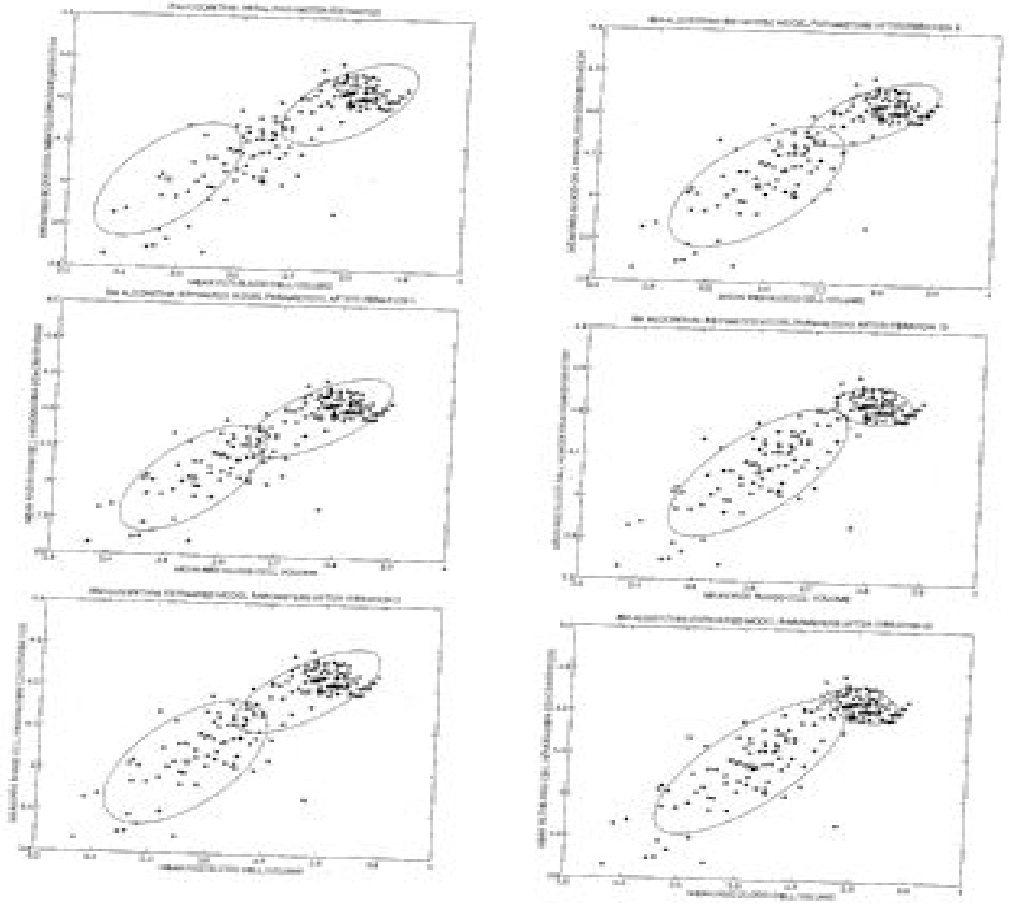
Wahrscheinlichkeitsbasiertes Clustering

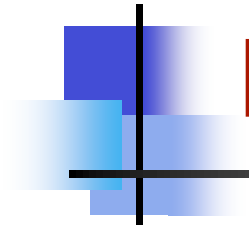
- Vorteile:
  - Das Wahrscheinlichkeitsmodell liefert eine total verteilungsbasierte Beschreibung für jede Komponente (→ unterschiedliche Cluster-Formen)
  - Gute Möglichkeit: (k+1)-te Komponente einführen als „Garbage Collector"

- Nachteile:
  - Modell beruht auf Annahmen
  - Hohe Komplexität der Schätz-Algorithmen (EM)
  - Meist Hintergrundwissen über die Daten erforderlich, um eine gute Wahl für die Verteilung treffen zu können

Wahrscheinlichkeitsbasiertes Clustering

- Diagramme (Bsp)





Fazit

- Schlecht vorhersehbare Ergebnisse
- Gutes oder schlechtes Ergebnis? → meist subjektive Einschätzung und anwendungsabhängig
- Oft ungenau / auch unsinnige Aufteilungen (Textmining)

- Applet für K-means:
  http://www.delft-cluster.nl/textminer/theory/kmeans/kmeans.html