7. Clusteranalyse

(= Häufungsanalyse; Clustering-Verfahren)

wird der multivariaten Statistik zugeordnet

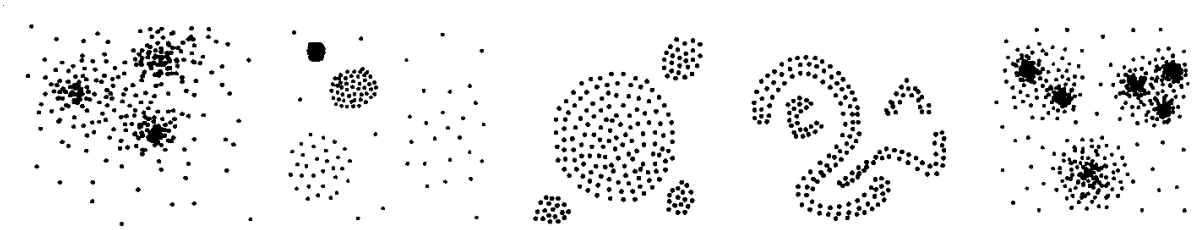
Voraussetzung wieder: Datenraum mit Instanzen, mehrere Attribute
- kein ausgezeichnetes Zielattribut, keine vorgegebenen Klassen

- Mittels Clusteranalyse sollen Strukturen (Gruppen, Cluster) in multidimensionalen Datensätzen gefunden werden.
- Clusteranalyse-Techniken sind nicht-überwachte Lernverfahren, bei denen ein Datensatz klassifiziert wird, ohne die Klassen vorher zu kennen.
- Die meisten Clusteringverfahren basieren auf einem Distanz- oder Unähnlichkeitsmaß zwischen den Daten.

(Klawonn 2004)

- Clusteranalyse-Verfahren sind den unüberwachten maschinellen Lernverfahren zuzuordnen.
- Eine Vielzahl von Clusteranalyseverfahren ist seit ca. 1960 entwickelt worden, dabei sind viele Verfahren erst mit zunehmender Hardwareleistung effizient anwendbar geworden.
- Ziel der Clusteranalyse: Die Objekte in Cluster aufteilen. Wobei ein Cluster eine Teilmenge in der Grundgesamtheit ist, die auf der Basis eines a priori definierten Distanz- bzw. Ähnlichkeitsmaßes, aus ähnlichen Objekten zusammengesetzt ist.
- Anwendungen: Vektorquantisierung/Kompression von multidimensionalen Eingabevektoren, Dichteschätzung

Verschiedene Arten von Clustern denkbar:



Was soll als "Cluster" gelten? Wann ist ein Cluster "gut"?

Bewertungsfunktion

Um nun die Güte einer Clusterung $\mathcal{C} = \{C_1, \ldots, C_k\}$ zu quantifizieren, wird eine Bewertungsfunktion $D(\mathcal{C})$ festgelegt werden, die dann optimiert werden kann.

Formal etwa so:

$$\begin{aligned} D : P(k, G) &\rightarrow \mathbb{R}^+ \\ \mathcal{C} &\rightarrow D(\mathcal{C}) \end{aligned}$$

hierbei ist $P(k, G)$ die Menge der möglichen Clusterungen der Grundgesamtheit G in k Cluster.

Zunächst muss ein Ähnlichkeits-Begriff genau definiert werden, hierauf basiert letztlich eine Bewertungsfunktion.

Falls k nicht durch die Anwendung spezifiziert ist, müssen Clusteranalysen mit verschiedenen Werten für k durchgeführt werden.

Theoretischer Algorithmus

1. Wähle eine Bewertungsfunktion D.
2. Setze k.
3. Berechne die optimale Clusterung $\mathcal{C}^{opt}$ durch

$$\mathcal{C}^{opt} := \operatorname{argmin}\{D(\mathcal{C}) \mid \mathcal{C} \in P(k, G)\}$$

Für beliebig große n und k ist der Algorithmus nicht realisierbar, da die Zahl der möglichen Clusterungen rasch anwächst.

Es sei $s(k,n) := |P(k,n)|$ die Anzahl der Elemente aus $P(k,n)$, dann gilt

$$s(k,n) = \frac{1}{k!}\sum_{j=1}^{k}(-1)^{k-j}\binom{k}{j}j^n$$

(Stirlingsche Zahlen 2. Art).

Das Optimum kann nur für kleine n und k durch vollständige Enumeration ermittelt werden.

Beispiel:

- $n = 20$ und $k = 4$, dann ist $s(k,n) = 45232115901$.
- $n = 100$ und $k = 5$, dann ist $s(k,n) > 10^{68}$.

Im allgemeinen Fall ist das Auffinden einer optimalen Clusterzerlegung ein NP-vollständiges Problem (Brucker 1974).

Voraussetzung für Clusterdefinition:
Distanz- bzw. *Ähnlichkeitsmaß*

Distanz- und Ähnlichkeitsmaße

Es sei $G \subset X$. Dann heißt $p : X \times X \to \mathbb{R}_+$ eine Distanz- bzw. Ähnlichkeitsfunktion auf X wenn gilt:

1. $p(x,x) = 0$ für alle $x \in X$. (zur Distanzfunktion)
   $p(x,x) \geq \max_y p(x,y)$ für alle $x \in X$. (zur Ähnlichkeitsfunktion)
2. $p(x,y) = p(y,x)$ für alle $x, y \in X$ (Symmetrie).
3. $p(x,y) \geq 0$ für alle $x, y \in X$.

Falls p eine Distanzfunktion ist, so heißt p eine **Metrik** auf X und (X,p) ein **metrischer Raum** falls außerdem gilt:

4. $p(x,y) = 0$, gdw. $x = y$.
5. $p(x,z) \leq p(x,y) + p(y,z)$ (Dreiecksungleichung).

(Schwenker 2004)

Gängige Distanzfunktionen:

1. Für Datensätze $x = (x_1, ..., x_d)$ mit *numerischen Attributwerten* x_i:

   Euklidische Distanz: $dist(x, y) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}$,

   Manhattan-Distanz: $dist(x, y) = |x_1 - y_1| + \ldots + |x_n - y_n|$,

   Maximums-Metrik: $dist(x, y) = max(|x_1 - y_1|, \ldots, |x_n - y_n|)$,

   Allgemeine L_p-Metrik: $dist(x, y) = \sqrt[p]{\sum_{i=1}^{d} (x_i - y_i)^p}$.

2. Für Datensätze $x = (x_1, ..., x_d)$ mit *kategorischen Attributwerten* x_i:

   Anzahl der verschiedenen Komponenten in x und y: $dist(x, y) = \sum_{i=1}^{d} \delta(x_i, y_i)$,

   wobei $\delta(x_i, y_i) = \begin{cases} 0 \text{ wenn } (x_i = y_i), \\ 1 \text{ wenn } (x_i \neq y_i). \end{cases}$

3. Für *endliche Mengen* $x = \{x_1, ..., x_d\}$:

   Anteil der verschiedenen Elemente in x und y: $dist(x, y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|}$.

3a. Speziell für **B** = $\{0; 1\}^n$:

- die L_1-Metrik ist auch auf **B** eine Metrik, die sogenannte *Hamming-Metrik*.
- Das Skalarprodukt ist eine Ähnlichkeitsfunktion auf **B**:

$$s(x, y) := \langle x, y \rangle := \sum_{i=1}^{n} x_i y_i$$

weitere Ähnlichkeitsmaße auf **B** siehe unten
(sog. Matching-Koeffizienten)

4. Für *Textdokumente*:
   Gegeben sei ein Vokabular T aus Termen t_i. Ein Dokument D wird dann repräsentiert durch $r(D) = \{f(t_i, D) \mid t_i \in T\}$, wobei $f(t_i, D)$ die Häufigkeit des Terms t_i im Dokument D ist. Ferner sei g eine monotone Dämpfungsfunktion wie die Quadratwurzel oder der Logarithmus, die komponentenweise angewendet wird. Die Distanz zwischen Dokumenten wird dann mit Hilfe der Vektoren $g(r(D)) = \{g(f(t_i, D))\}_{t_i \in T}$ definiert durch den Cosinus des Winkels zwischen den Dokumentvektoren: $dist(D_1, D_2) = 1 - \frac{\langle g(c(D_1)), g(c(D_2)) \rangle}{\|g(c(D_1))\| \cdot \|g(c(D_2))\|}$.
   Dabei ist $\langle \cdot , \cdot \rangle$ ein Skalarprodukt und $\| \cdot \|$ die damit definierte Länge von Vektoren.

(Ester & Sander 2000)

Matching-Koeffizienten:

Seien nun die **Merkmale binär**. Erweiterung auf nominalskalierte Merkmale ist einfach möglich.

Merkmalsausprägungen sind 0 und 1.

Ähnlichkeitsmaß zwischen $x, y \in \mathbb{B}^n$ sind durch sogenannte Vierfeldertafeln oder Kontingenztafeln definiert:

(x, y)	0	1
0	a_{00}	a_{01}
1	a_{10}	a_{11}

Dann gilt: $n = a_{00} + a_{01} + a_{10} + a_{11}$.

Gebräuchliche **matching-Koeffizienten** sind:

1. Simple-matching-coefficient (SMC)

$$d(x,y) = \frac{a_{00} + a11}{a_{00} + a_{01} + a_{10} + a_{11}} = \frac{a_{00} + a11}{n}$$

2. Jaccard-coefficient (JC)

$$d(x,y) = \frac{a11}{a_{01} + a_{10} + a_{11}} = \frac{a11}{n - a_{00}}$$

3. Rao-Russel-coefficient (RRC)

$$d(x,y) = \frac{a_{11}}{n}$$

Beispiel:

$x = (0,1,1,0,0,1,0,0,1,0,0,1,1,1,0,0,1,0,1,0)$
$y = (0,1,1,1,0,0,0,0,1,1,1,1,1,1,0,1,1,0,1,0)$
Dann ist $a_{00} = 7$, $a_{11} = 8$, $a_{10} = 1$ und $a_{01} = 4$ und es sind $d_{\text{SMC}} = \frac{15}{20}$, $d_{\text{JC}} = \frac{8}{13}$ und $d_{\text{RRC}} = \frac{8}{20}$.

Gemischte Merkmale

In praktischen Anwendungen kommen häufig metrische und nominal skalierte Merkmale vor.

Idee: Berechne Distanz-/Ähnlichkeitsmaß auf den metrischen und den nominal skalierten Merkmale getrennt.

$p_m(x_m, y_m)$ sei das Distanz-/Ähnlichkeitsmaß zwischen den metrisch skalierten Teilvektoren x_m und y_m von x bzw. y.

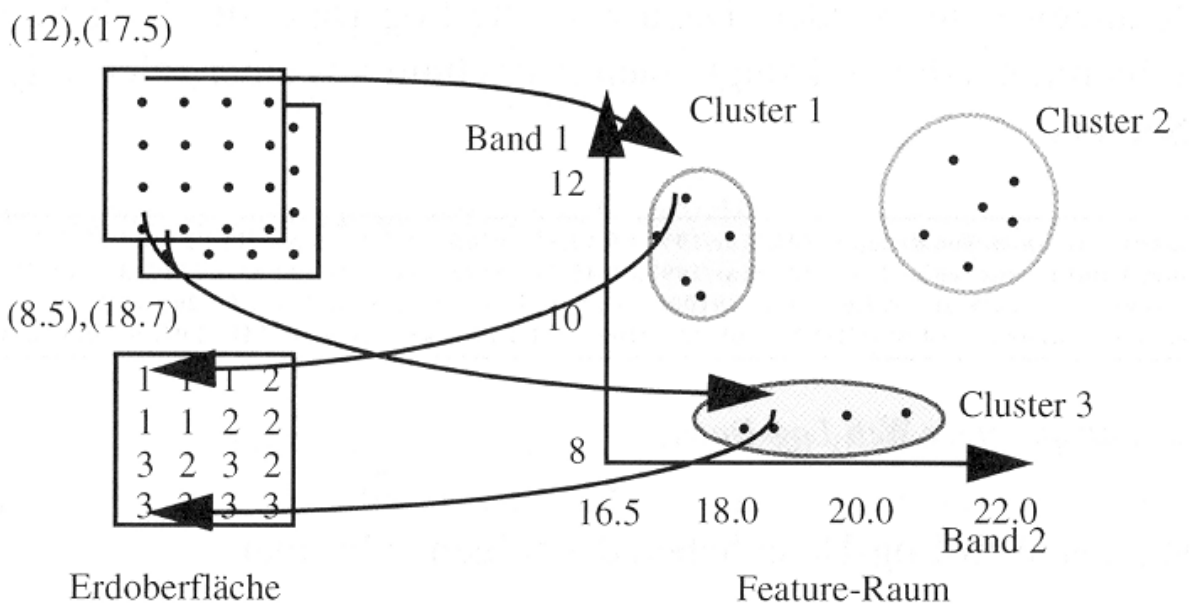
$p_n(x_n, y_n)$ sei das Distanz-/Ähnlichkeitsmaß zwischen den nominal skalierten Teilvektoren x_n und y_n von x bzw. y.

Dann definiere das Gesamt-Distanz-/Ähnlichkeitsmaß durch

$$p(x,y) = \alpha p_m(x_m, y_m) + (1 - \alpha) p_n(x_n, y_n)$$

Anwendungsbeispiele der Clusteranalyse:

- Clustering von Web-Sessions zur Bestimmung von Benutzergruppen
- Clustering von Sequenzen von Jahrringbreiten in der Paläo-klimatologie
- Clustering von Bildern in einer Bilddatenbank (um thematische Abfragen zu unterstützen)
- Clustering von DNA-Sequenzen verschiedener Organismen, um Verwandtschaftsgrade und Abstammungsbeziehungen zu klären (phylogenetische Analyse)
- Erstellung von thematischen Karten aus Satellitenbildern (Clustering wird in einem Merkmalsraum durchgeführt, wo jede Koordinate des aufgenommenen Gebietes durch einen Featurevektor repräsentiert wird)



(aus Ester & Sander 2000)

Durchführung der Clusteranalyse:

Die paarweisen Distanzen oder Ähnlichkeiten zwischen 2 Datenobjekten werden oft schon vorher berechnet und in einer Distanzmatrix *D* (bzw. Ähnlichkeitsmatrix *A*) abgespeichert.

Gegeben sei nun n Objekte in der Grundgesamtheit $G = \{e_1, \ldots, e_n\}$ und eine **Distanzmatrix**

$$D = (d_{\mu\nu})_{1 \leq \mu,\nu \leq n}$$

wobei $d_{\mu\nu}$ der Wert der Distanz zwischen e_μ und e_ν also etwa

$$d_{\mu\nu} = d(x_\mu, x_\nu)$$

für eine Distanzfunktion d und den Merkmalsvektoren x_μ und x_ν der Objekte e_μ bzw. e_ν.

Die vorgestellten Verfahren können leicht für Ähnlichkeitsfunktionen formuliert werden.

d braucht keine Metrik zu sein, also Dreiecksungleiung muss nicht gelten, ferner seien auch Nulleinträge ausserhalb der Diagonalen zulässig (d.h. aus $d(x, y) = 0$ muss nicht $x = y$ folgen).

(Schwenker 2004)

Überblick über Clustering-Verfahren:

Partitionierende Verfahren

- Parameter: Anzahl k der Cluster, Distanzfunktion
- sucht ein „flaches" Clustering in k Cluster mit minimalen Kosten

Hierarchische Verfahren

- Parameter: Distanzfunktion für Punkte und für Cluster
- bestimmt Hierarchie von Clustern, mischt jeweils die ähnlichsten Cluster

Dichtebasierte Verfahren

- Parameter: minimale Dichte in einem Cluster, Distanzfunktion
- erweitert Punkte um ihre Nachbarn solange Dichte groß genug

Andere Clustering-Verfahren

- Fuzzy Clustering
- Graph-theoretische Verfahren
- neuronale Netze

Partitionierende Verfahren:

zerlegen die Datenmenge in *k* Cluster

- *k* vorgegeben
- jedes Cluster enthält mindestens 1 Objekt
- jedes Objekt gehört zu genau einem Cluster
  (*Partition* der Datenmenge in Cluster)

Vorgehensweise:

- wähle *k* initiale Cluster-*Repräsentanten*
- optimiere diese iterativ
- ordne jedes Objekt seinem ähnlichsten ("nächsten") Repräsentanten zu

Mögliche Typen von Repräsentanten:

- Mittelwert des Clusters (*Konstruktion zentraler Punkte*)
- Element des Clusters (*Auswahl repräsentativer Punkte*)
- Wahrscheinlichkeitsverteilung für das Cluster (*Erwartungsmaximierungs-Verfahren*)

Konstruktion zentraler Punkte:

Grundbegriffe [Forgy 1965]

- Objekte sind Punkte $p=(x^p_1, ..., x^p_d)$ in einem euklidischen Vektorraum
- euklidische Distanz
- *Zentroid* μ_C: Mittelwert aller Punkte im Cluster C
- *Maß für die Kosten* (Kompaktheit) *eines Clusters C*

$$TD^2(C) = \sum_{p \in C} dist(p, \mu c)^2$$

- *Maß für die Kosten* (Kompaktheit) *eines Clustering*

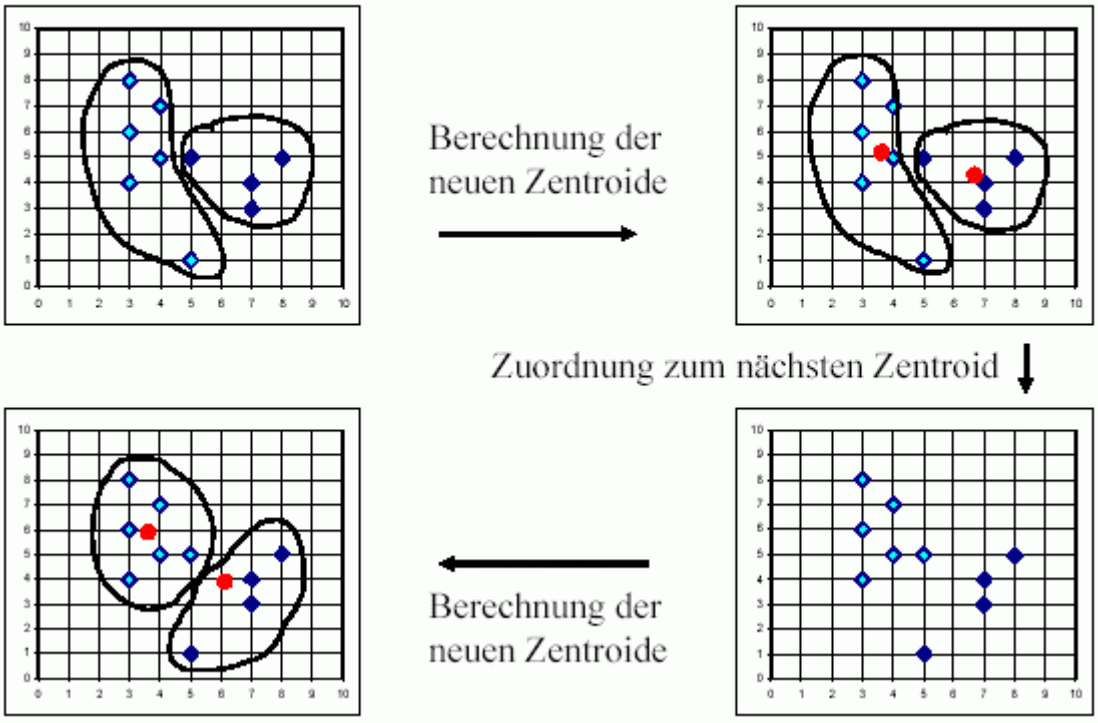
$$TD^2 = \sum_{i=1}^{k} TD^2(C_i)$$

Idee des Algorithmus

- Algorithmus startet z.B. mit zufällig gewählten Punkten als Cluster- Repräsentanten
- Der Algorithmus besteht aus zwei alternierenden Schritten:
  - Zuordnung jedes Datenpunktes zum räumlich nächsten Repräsentanten
  - Neuberechnung der Repräsentanten (Zentroid der zugeordneten Punkte)
- Diese Schritte werden so lange wiederholt, bis sich keine Änderung mehr ergibt

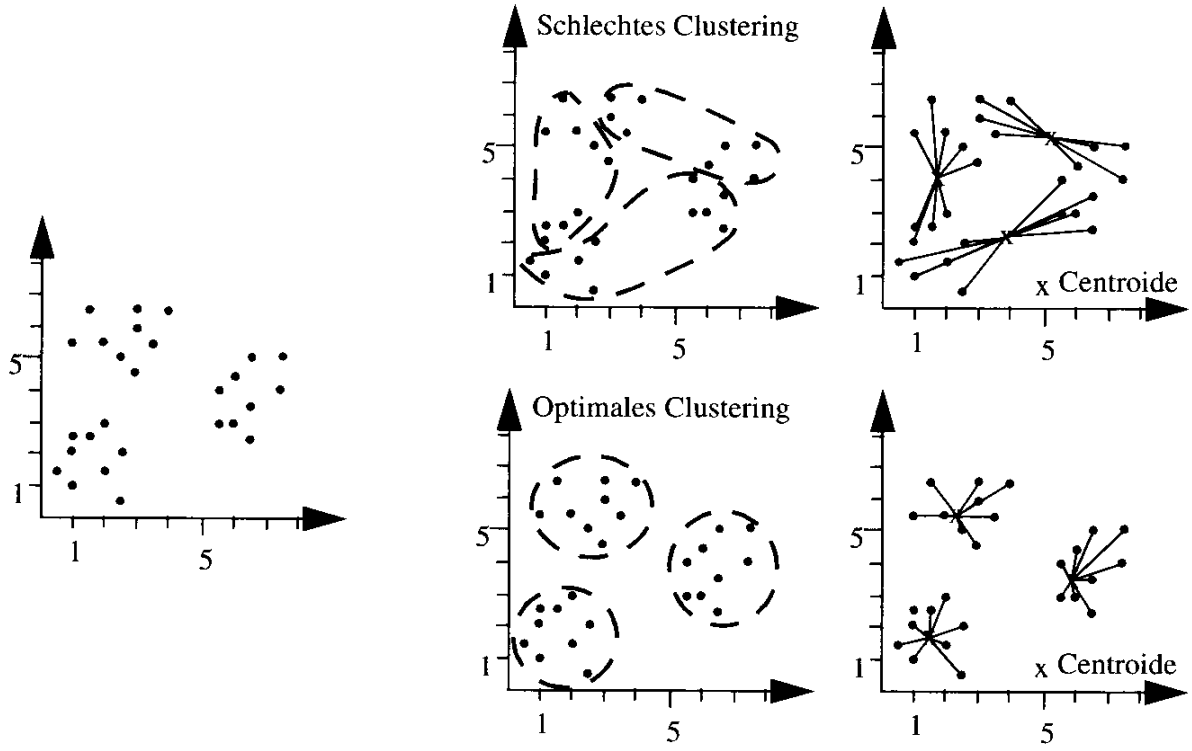
```
ClusteringDurchVarianzMinimierung(Punktmenge D, Integer k)
   Erzeuge eine „initiale“ Zerlegung der Punktmenge D in k
     Klassen;
   Berechne die Menge C'={C_1, ..., C_k} der Centroide für die k
     Klassen;
   C = {};
   repeat until C = C'
      C = C';
      Bilde k Klassen, durch Zuordnung jedes Punktes zum
        nächstliegenden Centroid aus C; // Schritt 1
      Berechne die Menge C'={C'_1, ..., C'_k} der Centroide für
        die neu bestimmten Klassen; // Schritt 2
   return C;
```

Beispiel

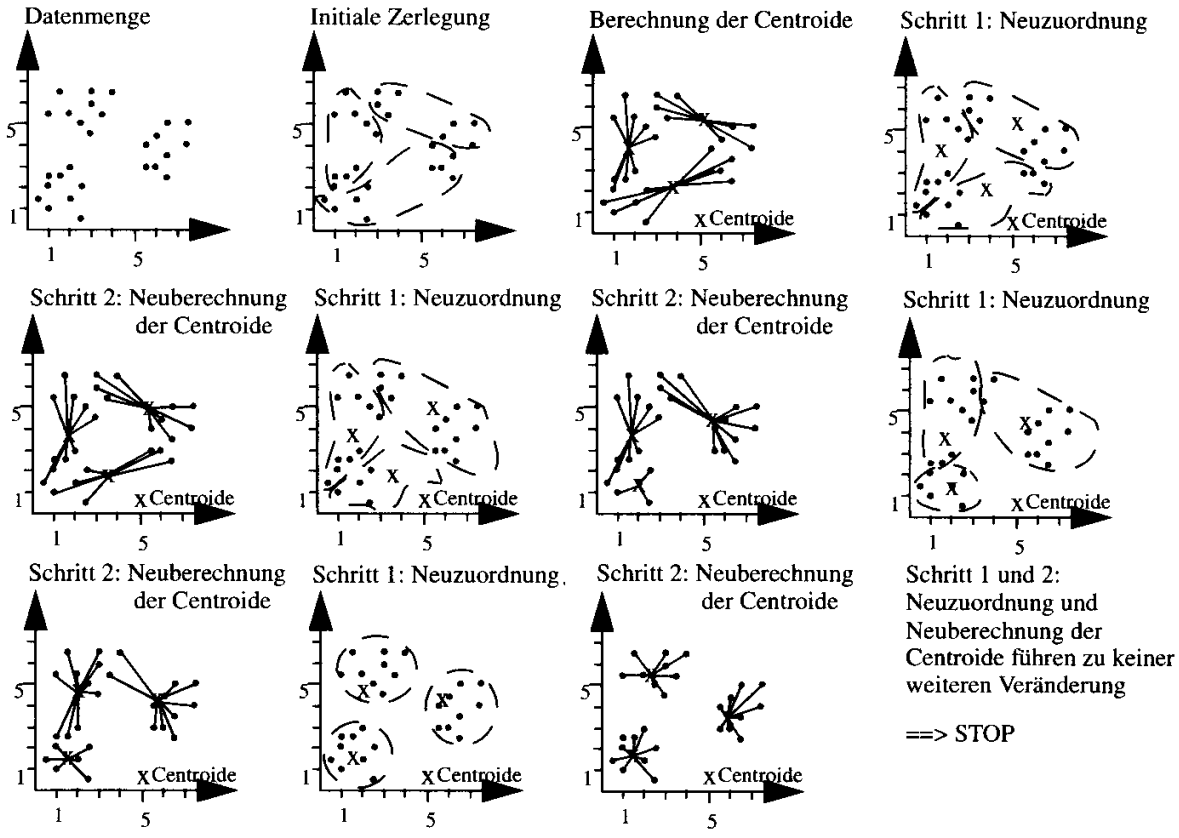


(Böhm 2003)

Beispiel mit $k = 3$:



Durchführung des Verfahrens, beginnend mit der "schlechten" Zerlegung:



(aus Ester & Sander 2000)

Varianten des Basis-Algorithmus:

k-means-Algorithmus (MacQueen 1967)
- Idee: die betroffenen Zentroide werden sofort aktualisiert, wenn ein Punkt seine Clusterzugehörigkeit ändert
- hat im wesentlichen die Eigenschaften des Basisalgorithmus
- ist aber reihenfolgeabhängig

ISODATA
- basiert auf *k*-means
- Verbesserung des Ergebnisses durch Operationen wie
  - Elimination sehr kleiner Cluster
  - Verschmelzung und Aufspalten von Clustern
- Benutzer muß viele zusätzliche Parameter angeben

zum Verfahren "Konstruktion zentraler Punkte":

Diskussion

+ Effizienz
  Aufwand: O(n) für eine Iteration,
  Anzahl der Iterationen ist im allgemeinen klein (~ 5 - 10).
+ einfache Implementierung

➡ *K*-means ist das populärste partitionierende Clustering-Verfahren

– Anfälligkeit gegenüber Rauschen und Ausreißern
  alle Objekte gehen ein in die Berechnung des Zentroids
– Cluster müssen konvexe Form haben
– die Anzahl k der Cluster ist oft schwer zu bestimmen
– starke Abhängigkeit von der initialen Zerlegung
  sowohl Ergebnis als auch Laufzeit

Verfahren "Auswahl repräsentativer Punkte":

Grundbegriffe [Kaufman & Rousseeuw 1990]

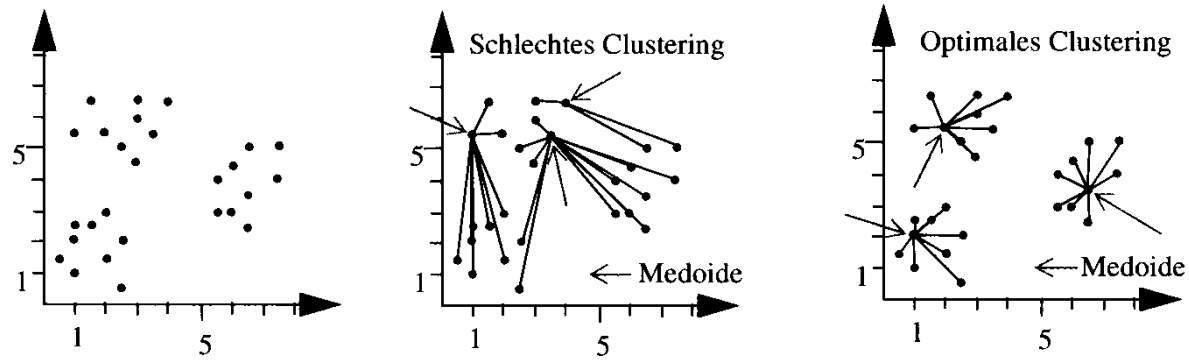
- setze nur Distanzfunktion für Paare von Objekten voraus
- *Medoid*: ein zentrales Element des Clusters (repräsentativer Punkt)
- *Maß für die Kosten* (Kompaktheit) *eines Clusters C*

$$TD(C) = \sum_{p \in C} dist(p, mc)$$

- *Maß für die Kosten* (Kompaktheit) *eines Clustering*

$$TD = \sum_{i=1}^{k} TD(C_i)$$

die Laufzeitkomplexität der erschöpfenden Suche ist $O(n^k)$



Überblick über k-medoid Algorithmen

PAM [Kaufman & Rousseeuw 1990]

- Greedy-Algorithmus:
  in jedem Schritt wird nur ein Medoid mit einem Nicht-Medoid vertauscht
- vertauscht in jedem Schritt das Paar (Medoid, Nicht-Medoid), das die größte Reduktion der Kosten TD bewirkt

CLARANS [Ng & Han 1994]

zwei zusätzliche Parameter: *maxneighbor* und *numlocal*

- höchstens *maxneighbor* viele von zufällig ausgewählten Paaren (Medoid, Nicht-Medoid) werden betrachtet
- die erste Ersetzung, die überhaupt eine Reduzierung des *TD*-Wertes bewirkt, wird auch durchgeführt
- die Suche nach *k* „optimalen“ Medoiden wird *numlocal* mal wiederholt

Algorithmus PAM

```
PAM(Objektmenge D, Integer k, Float dist)
   Wähle das Objekt, für das TD(D) minimal ist, als ersten
     Medoid;
   for i=2 to k do
      wähle als i-ten Mediod das Objekt, welches den Wert für TD
        minimiert;
   TD_Änderung := −∞;
   while TD_Änderung < 0 do
      Berechne für jedes Paar (Medoid M, Nicht-Medoid N) den
        Wert TD_{N↔M}, d.h. den Wert TD unter der Annahme, daß der
        Medoid M durch den Nicht-Medoid N ersetzt wird;
      Wähle das Paar (M, N), für das TD_Änderung := TD_{N↔M} − TD
        minimal ist;
      if TD_Änderung < 0 then
         ersetze den Medoid M durch den Nicht-Medoid N;
         Speichere die aktuellen Medoide als die bisher beste
           Partitionierung;
   return Medoide;
```

Algorithmus CLARANS

```
CLARANS(Objektmenge D, Integer k, Real dist,
          Integer numlocal, Integer maxneighbor)
   TD_best := ∞;
   for r from 1 to numlocal do
      wähle zufällig k Objekte als Medoide und berechne TD;
      i := 0;
      while i < maxneighbor do
         Wähle zufällig ein Paar (Medoid M, Nicht-Medoid N);
         Berechne TD_Änderung := TD_{N↔M} − TD, wobei TD_{N↔M} der TD-
           Wert ist, der sich ergibt, wenn der Medoid M durch den
           Nicht-Medoid N ersetzt wird;
         if TD_Änderung < 0 then
            ersetze den Medoid M durch den Nicht-Medoid N;
            TD := TD_{N↔M};
            i := 0;
         else
            i:= i + 1;
      if TD < TD_best then
         TD_best := TD;
         Speichere die aktuellen Medoide als die bisher beste
           Partitionierung;
   return Medoide;
```

(aus Ester & Sander 2000)

Laufzeitvergleich (experimentelle Untersuchung):
CLARANS ist PAM deutlich überlegen, hat aber immer noch ungefähr quadratische Laufzeit (bezogen auf Zahl der Objekte)

Erwartungsmaximierung ("EM-Algorithmus"):

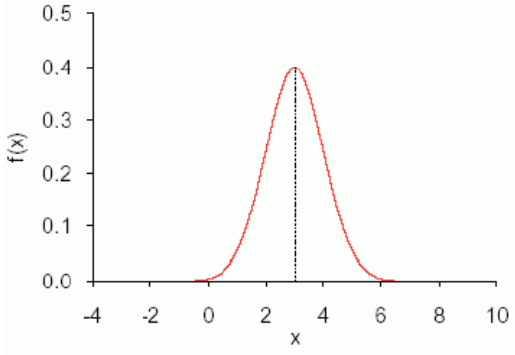
[Dempster, Laird & Rubin 1977]

- Objekte sind Punkte $p=(x^p_1, ..., x^p_d)$ in einem euklidischen Vektorraum
- ein Cluster wird durch eine Wahrscheinlichkeitsverteilung beschrieben
- typisch: Modell für einen Cluster ist eine multivariate Normalverteilung
- Repräsentation eines Clusters C
  - Mittelwert μ_C aller Punkte des Clusters
  - d x d Kovarianzmatrix Σ_C für die Punkte im Cluster C
- Wahrscheinlichkeitsdichte eines Clusters C

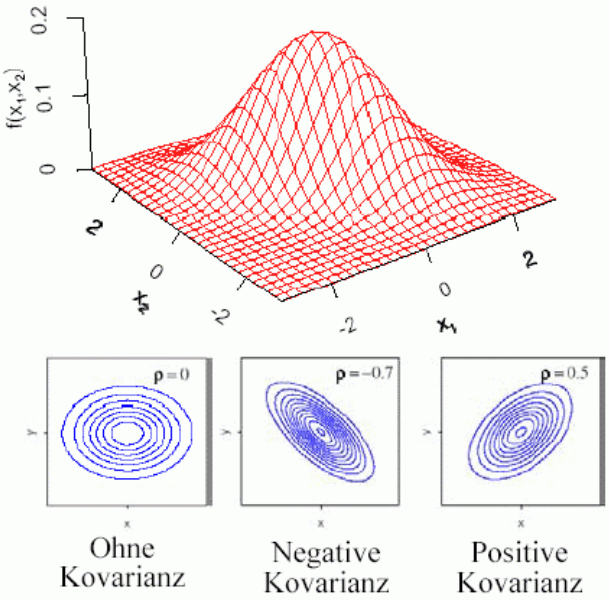
$$P(x \mid C) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_C \mid}} e^{\frac{1}{2}(x-\mu_C)^T \cdot (\Sigma_C)^{-1} \cdot (x-\mu_C)}$$

Multivariate Normalverteilung

Univariate Normalverteilung



Bivariate Normalverteilung



Erwartungsmaximierungs-Algorithmus

Idee:

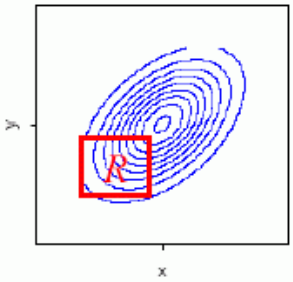
- Jeder Punkt gehört zu mehreren (eigentlich *allen*) Clustern, jeweils mit unterschiedlicher Wahrscheinlichkeit, abh. v. $P(x|C)$
- Algorithmus besteht wieder aus zwei alternierenden Schritten:
  - Zuordnung von Punkten zu Clustern (hier nicht absolut sondern relativ/nach Wahrscheinlichkeit)
  - Neuberechnung der Cluster-Repräsentanten (Gauß-Kurven)

Alles muss auf eine stochastische Grundlage gestellt werden:

- Bei Berechnung der Clusterzentren (μ_i) muss berücksichtigt werden, dass Punkte Clustern nicht absolut sondern nur relativ zugeordnet sind
- Wie groß ist die Wahrscheinlichkeit der Clusterzugehörigkeit?

Jeder Cluster C_i wird durch eine Wahrscheinlichkeitsdichte-Funktion (Normalverteilung) modelliert:

$$P(x \mid C_i) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_{Ci} \mid}} e^{\frac{1}{2}\cdot(x-\mu_{Ci})^T\cdot(\Sigma_{Ci})^{-1}\cdot(x-\mu_{Ci})}$$



Dichtefunktion:

- Integral über den Gesamtraum ergibt 1
- Integral über Region R ergibt Wahrscheinlichkeit, dass in der Region ein beliebiger Punkt des Clusters liegt, bzw. den relativen Anteil (z.B. 30%) der Punkte des Clusters, die in R liegen

$$P(x \mid C_i) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_{Ci} \mid}} e^{\frac{1}{2}\cdot(x-\mu_{Ci})^T\cdot(\Sigma_{Ci})^{-1}\cdot(x-\mu_{Ci})}$$

Bedingte Wahrscheinlichkeit:

- Dies würde unter der Voraussetzung gelten, dass der Punkt x ausschließlich dem Cluster C_i zugeordnet wäre (was nicht stimmt)
- Deshalb Notation als *bedingte* Wahrscheinlichkeit

Bei k Gaussverteilungen (durch k Cluster) ergibt sich folgende Gesamt-Wahrscheinlichkeitsdichte:

$$P(x) = \sum_{i=1}^{k} W_i \cdot P(x|C_i)$$

wobei W_i der relative Anteil der Datenpunkte ist, der zum Cluster C_i gehört (z.B. 5%), was man auch als Gesamt-Wahrscheinlichkeit des Clusters $P(C_i)$ interpretieren kann.

Mit dem *Satz von Bayes* kann man die Wahrscheinlichkeit bestimmen, dass ein gegebener Punkt x zum Cluster C_i gehört, geschrieben als bedingte Wahrscheinlichkeit $P(C_i|x)$

$$P(C_i|x) = W_i \cdot \frac{P(x|C_i)}{P(x)}$$

- Maß für die Güte eines Clustering M

$$E(M) = \sum_{x \in D} \log(P(x))$$

⇨ $E(M)$ soll maximiert werden.

- Anteil des Clusters an der Datenmenge:

$$W_i = P(C_i) = \frac{1}{n} \sum_{i=1}^{k} P(C_i \mid x)$$

- Mittelwert und Kovarianzmatrix der Gaußverteilung:

$$\mu_i = \left(\sum_{x \in D} x \cdot P(C_i \mid x)\right) / \left(\sum_{x \in D} P(C_i \mid x)\right)$$

$$\Sigma_i = \left(\sum_{x \in D} (x - \mu_i)(x - \mu_i)^T \cdot P(C_i \mid x)\right) / \left(\sum_{x \in D} P(C_i \mid x)\right)$$

(Böhm 2003)

Je größer der Wert für $E(M)$ ist, desto wahrscheinlicher sind die gegebenen Daten D unter der Annahme, dass sie durch Mischung von k Gaußverteilungen entstanden sind.

```
ClusteringDurchErwartungsmaximierung(Punktmenge D, Integer k)
  Erzeuge ein „initiales" Modell M' = (C1', ..., Ck') von
    Gaußverteilungen für D;
  repeat
    // Schritt 1 „Neuzuordnung"
    Berechne die oben definierten Wahrscheinlichkeiten
      P(x|Ci), P(x) und P(Ci|x) für jedes Objekt aus D und jede
      Gaußverteilung/jeden Cluster Ci;
    // Schritt 2 „Neuberechnung der Clusterrepräsentation"
    Berechne ein neues Modell M={C1, ..., Ck} von
      Gaußverteilungen durch Neuberechnung der Parameter Wi,
      μCi, ΣCi für jedes i = 1, ..., k;
    M' := M;
  until |E(M) - E(M')| < ε
  return M;
```

Die Parameter W_i, μ_{C_i} und Σ_{C_i} werden dabei folgendermaßen neuberechnet:

$$W_i = \frac{1}{n} \sum_{x \in D} P(C_i|x),$$

$$\mu_i = \frac{\sum_{x \in D} x \cdot P(C_i|x)}{\sum_{x \in D} P(C_i|x)},$$

$$\Sigma_i = \frac{\sum_{x \in D} P(C_i|x)(x-\mu_i)(x-\mu_i)^T}{\sum_{x \in D} P(C_i|x)}.$$

(Ester & Sander 2000)

Diskussion

- Aufwand:

  $O(n * |M| * \#Iterationen)$

  Anzahl der benötigten Iterationen im allgemeinen sehr hoch

- Ergebnis und Laufzeit hängen (wie beim *k-means* und *k-medoid*) stark ab
  - von der initialen Zuordnung
  - von der „richtigen" Wahl des Parameters *k*
- Modifikation für Partitionierung der Daten in *k disjunkte* Cluster:

  jedes Objekt nur demjenigen Cluster zuordnen,

  zu dem es am wahrscheinlichsten gehört.

(Böhm 2003)

Wahl des initialen Clustering:

Idee

- Clustering einer kleinen Stichprobe liefert im allgemeinen gute initiale Cluster
- einzelne Stichproben sind evtl. deutlich anders verteilt als die Grundgesamtheit

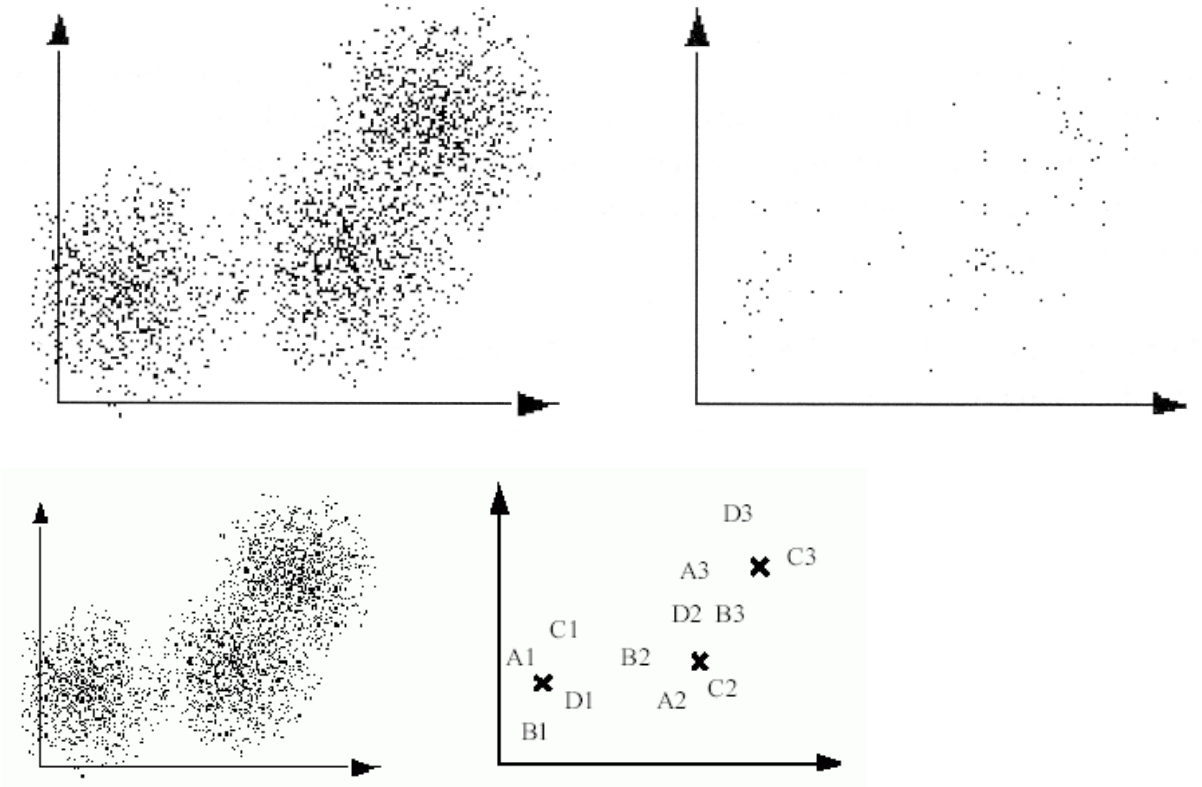
Methode [Fayyad, Reina & Bradley 1998]

- ziehe unabhängig voneinander m verschiedene Stichproben
- clustere jede der Stichproben

  ⇨ m verschiedene Schätzungen für k Clusterzentren

  $A = (A_1, A_2, \ldots, A_k), B = (B_1, \ldots, B_k), C = (C_1, \ldots, C_k), \ldots$
- Clustere nun die Menge $DB = A \cup B \cup C \cup \ldots$
  mit m verschiedenen Stichproben $A, B, C, \ldots$ als Startkonfiguration
- Wähle von den m Clusterings dasjenige mit dem besten Wert
  bezüglich des zugehörigen Maßes für die Güte eines Clustering

Beispiel:
3 Gaußcluster in der Ebene: Gesamtmenge und Stichprobe



Grundgesamtheit

$k = 3$ Gauß-Cluster

DB

von $m = 4$ Stichproben

✖ *wahre Clusterzentren*

Wahl des Parameters *k* (für alle bisher vorgestellten Verfahren):

Methode

- Bestimme für $k = 2, ..., n\text{-}1$ jeweils ein Clustering
- Wähle aus der Menge der Ergebnisse das „beste“ Clustering aus

Maß für die Güte eines Clusterings

- muß unabhängig von der Anzahl k sein
- bei k-means und k-medoid: TD^2 und TD sinken monoton mit steigendem k
- bei EM: E steigt monoton mit steigendem k

Brauche ein von k unabhängiges Gütemaß für die k-means- und k-medoid-Verfahren

➡ *Silhouetten-Koeffizient*

Silhouetten-Koeffizient [Kaufman & Rousseeuw 1990]

- sei $a(o)$ der Abstand eines Objekts o zum Repräsentanten seines Clusters und $b(o)$ der Abstand zum Repräsentanten des „zweitnächsten“ Clusters
- Silhouette $s(o)$ von o

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$

$$-1 \leq s(o) \leq +1$$

$s(o) \approx -1 / 0 / +1$: schlecht / indifferent / gute Zuordnung

Silhouettenkoeffizient s_C eines Clustering

durchschnittliche Silhouette aller Objekte

- Interpretation des Silhouettenkoeffizients

$s_C > 0{,}7$: starke Struktur,

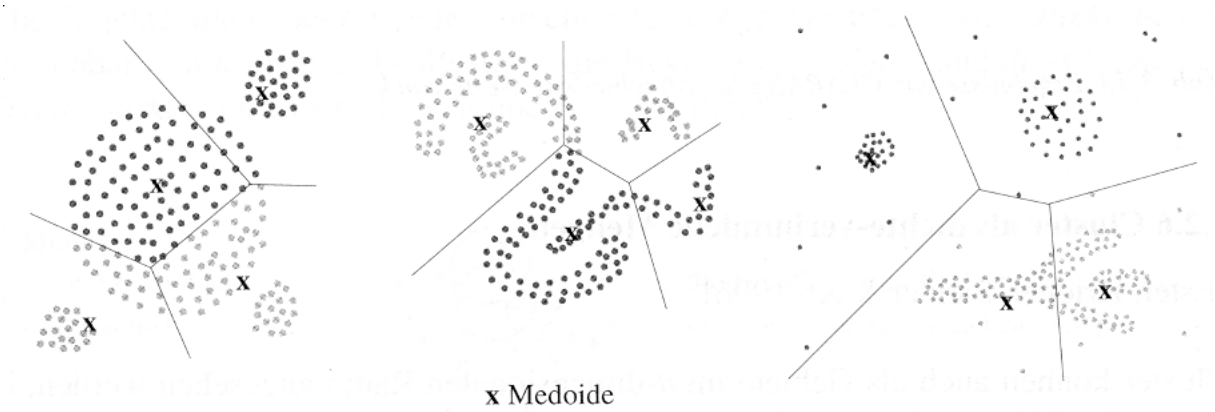
$s_C > 0{,}5$: brauchbare Struktur, . . .

(Böhm 2003)

Probleme bei iterativ arbeitenden Clustering-Verfahren:
arbeiten nicht so gut, wenn

- Cluster nicht kugelförmig / elliptisch sind
- stark unterschiedliche Größe haben
- stark unterschiedliche Punktdichten haben

Beispiel: Ergebnisse von CLARANS (mit k = 4):



(Ester & Sander 2000)

$\Rightarrow$ verwende *dichtebasierte* Verfahren

Cluster als dichte-verbundene Mengen

Grundlagen

Idee

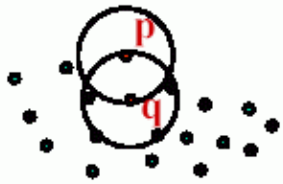
- Cluster als Gebiete im d-dimensionalen Raum, in denen die Objekte dicht beieinander liegen
- getrennt durch Gebiete, in denen die Objekte weniger dicht liegen

Anforderungen an dichtebasierte Cluster

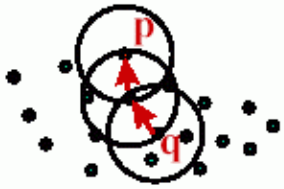
- für jedes Objekt eines Clusters überschreitet die lokale Punktdichte einen gegebenen Grenzwert
- die Menge von Objekten, die den Cluster ausmacht, ist räumlich zusammenhängend

Grundbegriffe [Ester, Kriegel, Sander & Xu 1996]

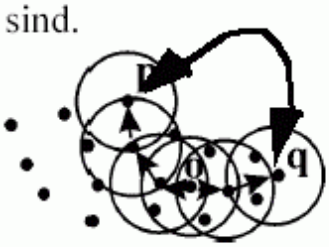
- Ein Objekt o $\in O$ heißt *Kernobjekt*, wenn gilt:
  $|N_\varepsilon(o)| \geq MinPts$, wobei $N_\varepsilon(o) = \{o' \in O \mid dist(o, o') \leq \varepsilon\}$.



- Ein Objekt $p \in O$ ist *direkt dichte-erreichbar* von $q \in O$ bzgl. ε und *MinPts*, wenn gilt: $p \in N_\varepsilon(q)$ und q ist ein Kernobjekt in O.
- Ein Objekt p ist *dichte-erreichbar* von q, wenn es eine Kette von direkt erreichbaren Objekten zwischen q und p gibt.



- Zwei Objekte p und q sind *dichte-verbunden*, wenn sie beide von einem dritten Objekt o aus dichte-erreichbar sind.



- Ein *Cluster* C bzgl. ε und *MinPts* ist eine nicht-leere Teilmenge von O mit für die die folgenden Bedingungen erfüllt sind:

  Maximalität: $\forall p,q \in O$: wenn $p \in C$ und q dichte-erreichbar von p ist, dann ist auch $q \in C$.

  Verbundenheit: $\forall p,q \in C$: p ist dichte-verbunden mit q.

- Definition Clustering

  Ein *dichte-basiertes Clustering CL* der Menge O bzgl. ε und *MinPts* ist eine „vollständige" Menge von dichte-basierten Clustern bzgl. ε und *MinPts* in O.

- Dann ist die Menge $Noise_{CL}$ („*Rauschen*") definiert als die Menge aller Objekte aus O, die nicht zu einem der dichte-basierten Cluster C_i gehören.

- Grundlegende Eigenschaft

  Sei C ein dichte-basierter Cluster und sei $p \in C$ ein Kernobjekt. Dann gilt: $C = \{o \in O \mid o$ dichte-erreichbar von p bzgl. ε und *MinPts*$\}$.

(Böhm 2003)

Algorithmus DBSCAN („Density-Based Clustering of Applications with Noise")

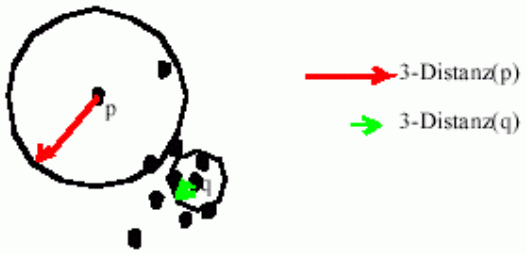
```
DBSCAN(Objektmenge D, Real ε, Integer MinPts)
   // Zu Beginn sind alle Objekte unklassifiziert, d.h.
   // o.ClId = UNKLASSIFIZIERT für alle o ∈ Objektmenge
   ClusterId := nextId(NOISE);
   for i from 1 to |D| do
      Objekt := D.get(i);
      if Objekt.ClId = UNKLASSIFIZIERT then
         if ExpandiereCluster(D, Objekt, ClusterId, ε, MinPts)
           then ClusterId:=nextId(ClusterId);

ExpandiereCluster(Objektmenge D, Objekt StartObjekt,
          Integer ClusterId, Real ε, Integer MinPts): Boolean;
   seeds := N_ε(StartObjekt);
   if |seeds| < MinPts then // StartObjekt ist kein Kernobjekt
      StartObjekt.ClId := NOISE;
      return false;
   // sonst: StartObjekt ist ein Kernobjekt
   for each o ∈ seeds do o.ClId := ClusterId;
   entferne StartObjekt aus seeds;
   while seeds ≠ ∅ do
      wähle ein Objekt o aus der Menge seeds;
      Nachbarschaft := N_ε(o);
      if |Nachbarschaft| ≥ MinPts then // o ist ein Kernobjekt
         for i from 1 to |Nachbarschaft| do
            p := Nachbarschaft.get(i);
            if p.ClId in {UNKLASSIFIZIERT, NOISE} then
               if p.ClId = UNKLASSIFIZIERT then
                  füge p zur Menge seeds hinzu;
               p.ClId := ClusterId;
      entferne o aus der Menge seeds;
   return true;
```

(Ester & Sander 2000)

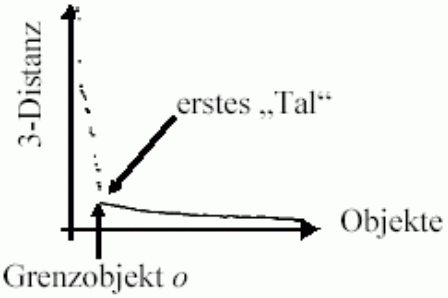
Parameterbestimmung

- Cluster: Dichte größer als die durch ε und *MinPts* spezifizierte „Grenzdichte“
- Gesucht: der am wenigsten dichte Cluster in der Datenmenge
- Heuristische Methode: betrachte die Distanzen zum *k*-nächsten Nachbarn.



- Funktion *k-Distanz*: Distanz eines Objekts zu seinem *k*-nächsten Nachbarn
- *k-Distanz-Diagramm*: die *k*-Distanzen aller Objekte absteigend sortiert

Beispiel eines *k*-Distanz-Diagramms

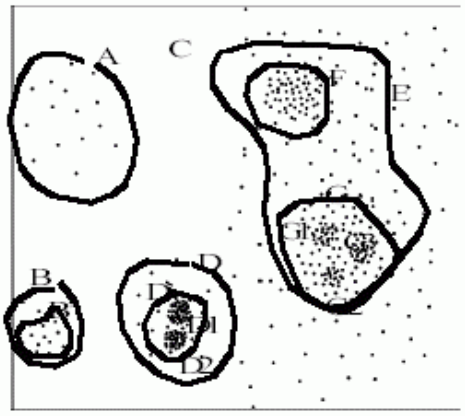


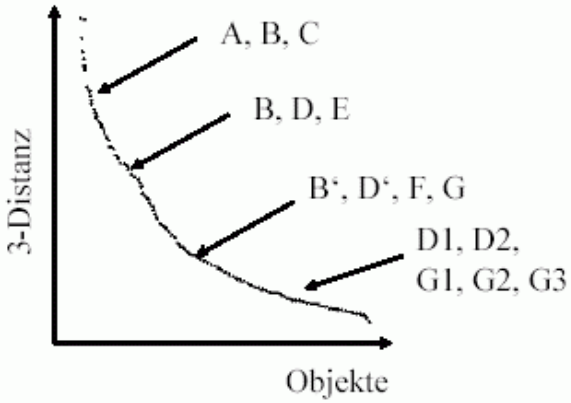
Heuristische Methode

- Benutzer gibt einen Wert für *k* vor (Default ist $k = 2*d - 1$), $MinPts := k+1$.
- System berechnet das *k*-Distanz-Diagramm und zeigt das Diagramm an.
- Der Benutzer wählt ein Objekt *o* im *k*-Distanz-Diagramm als Grenzobjekt aus, $\varepsilon :=$ *k*-Distanz(*o*).

Probleme der Parameterbestimmung

- hierarchische Cluster
- stark unterschiedliche Dichte in verschiedenen Bereichen des Raumes
- Cluster und Rauschen sind nicht gut getrennt





(Böhm 2003)

$\Rightarrow$ verwende *hierarchische* Clustering-Verfahren

Grundlage: Verfeinerungs- / Vergröberungsrelation zwischen verschiedenen Partitionen derselben Menge

Eine Partition $\mathcal{C}$ auf G ist eine Menge $\mathcal{C} = \{C_1, \ldots, C_k\}$ mit

1. $C_i \neq \emptyset$ für alle i
2. $C_i \cap C_j = \emptyset$ für alle i und j mit $i \neq j$.
3. $C_1 \cup \cdots \cup C_k = G$

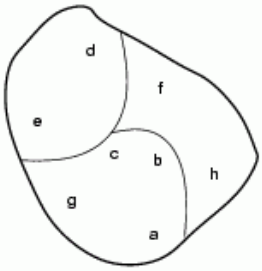
Eine Partition $\mathcal{B}$ heißt eine **Verfeinerung** einer Partition $\mathcal{C}$ falls jede Menge $B_i \in \mathcal{B}$ Teilmenge geneu einer Menge $C_j \in \mathcal{C}$ ist.
Notation: $\mathcal{B} \subset \mathcal{C}$.

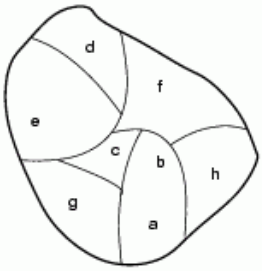
Beispiel:

$\mathcal{C} = \{\{a, b, c, g\}, \{d, e\}, \{f, h\}\}$

$\mathcal{B} = \{\{a, b\}, \{c\}, \{g\}, \{d\}, \{e\}, \{f\}, \{h\}\}$

Dann ist $\mathcal{B} \subset \mathcal{C}$.





Agglomerative und divisive Verfahren

Bei einer hierarchischen Clusteranalyse entsteht eine Folge von Partitionen der Grundmenge G, in der Folge werden die Partitionen monoton verfeinert bzw. vergröbert (je nach Verfahren).

Man unterscheidet bei den hierarchischen Clusterverfahren

- **agglomerative (aufbauende) Clusterverfahren** (Partiionen werden im Verlauf der Clusteranalyse gröber)
- **divisive (teilende) Clusterverfahren** (Partionen werden im Verlauf der Clusteranalyse feiner)
- Agglomerative Clusterverfahren starten mit der Anfangsclusterung

  $$\mathcal{C}_1 = \{\{e_1\}, \{e_2\}, \ldots, \{e_n\}\}$$

  dies ist offenbar die feinste Partition von G und terminieren mit

  $$\mathcal{C}_n = \{\{e_1, \ldots, e_n\}\}$$

  der gröbsten Partition von G.

  Im Verlauf der Clusteranalyse werden jeweils zwei Cluster C_i und C_j zu einem Cluster C_f vereinigt. Die Auswahl der beiden Cluster C_i und C_j geschieht auf der Basis der Distanzmatrix D.

  Es entsteht so eine Folge von Clusterungen $\mathcal{C}_{i=1}^{n}$ von der G mit der Eigenschaft: $\mathcal{C}_{i-1} \subset \mathcal{C}_i$.

- Die divisiven Clusterverfahren gehen genau anders vor, sie starten mit

  $$\mathcal{C}_0 = \{\{e_1, \ldots, e_n\}\}$$

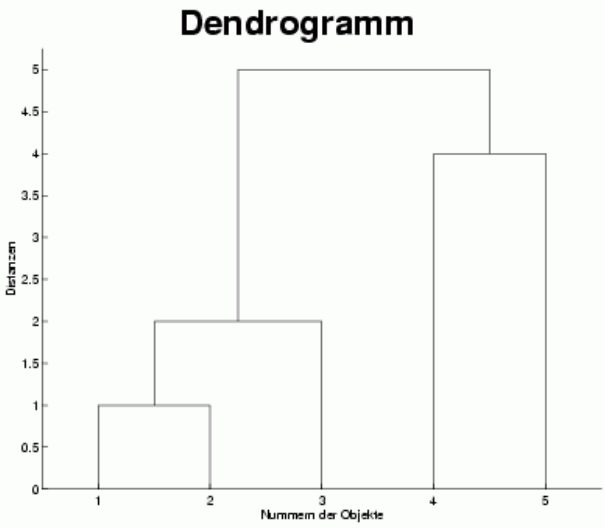
  und wählen in jedem Iterationsschritt i ein Cluster $C \in \mathcal{C}_s$ nach einem vordefinierten Kriterium aus und teilen es dann in 2 Cluster C_i und C_j auf.

Dendrogramme

Dendrogramme sind zur graphischen Dastellung von Folgen hierarchischer Clusterungen

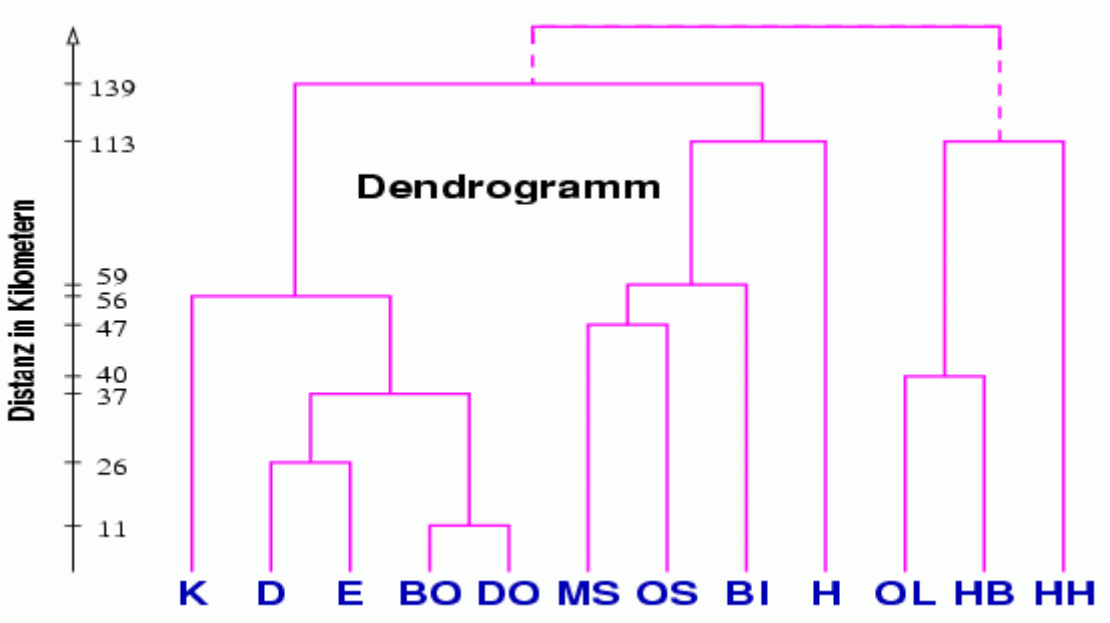
Clusterungen

1. $\{\{1,2,3,4,5\}\}$
2. $\{\{1,2,3\},\{4,5\}\}$
3. $\{\{1,2,3\},\{4\},\{5\}\}$
4. $\{\{1,2\},\{3\},\{4\},\{5\}\}$
5. $\{\{1\},\{2\},\{3\},\{4\},\{5\}\}$



(Schwenker 2004)

Beispiel:



(Schukat-Talamazzini 2002)

Agglomerative Clusteranalyse

1. Eingabe ist eine $n \times n$ Distanzmatrix $D = (d_{ij})$ der Grundmenge $G = \{e_1, \ldots, e_n\}$. (In der Praxis wir nur die obere Dreiecksmatrix von D benutzt.)

2. Ausgabe eine Folge von Partitionen.

3. Start mit der feinsten Partition $\mathcal{C}_1 = \{\{e_1\}, \ldots, \{e_n\}\}$. Jedes Objekt e_i definiert also genau ein Cluster $C_i = \{e_i\}$.

4. Im Verlauf der Clusteranalyse werden in jedem Verarbeitungsschritt jeweils die beiden Cluster fusioniert, die die geringste Distanz haben.

5. Idee: Distanz-(funktion) d die zwischen den Objekten definiert ist, muss zu einer Distanz-(funktion) d_c zwischen den Clustern erweitert werden. Offenbar $d_c(\{e_i\}, \{e_j\}) := d(e_i, e_j)$.

Agglomerativer Basisalgorithmus

1. Input: $D = (d_{ij})$

2. Bestimme die beiden Cluster C_{i^*} und C_{j^*} mit der geringsten Distanz:

$$d_c(C_{i^*}, C_{j^*}) = \min d_c(C_i, C_j)$$

3. Fusion der Cluster C_{i^*} und C_{j^*}, d.h. streiche C_{i^*} und C_{j^*} und nehme dafür $C_F := C_{i^*} \cup C_{j^*}$ in die bisherige Clusterung auf:

4. Aktualisiere die Distanzmatrix D

   (a) Streiche die zu C_{i^*} und C_{j^*} gehörenden Zeilen und Spalten.
   (b) **Berechung der Distanzen** zwischen D_F und den verbleibenden Clustern C_r.

5. Stop falls gröbste Clusterung $\mathcal{C}_n = \{\{e_1, \ldots, e_n\}\}$ erreicht; sonst: goto 2.

Unterschied zwischen den verschiedenen hierarchischen-agglomerativen Verfahren ist nun der Berechnungsschritt 4 zur Neuberechnung der sogenannten **Inter-Cluster-Distanzen** zwischen D_F und den verbleibenden Clustern C_r.

Die hierarchischen-agglomerativen Verfahren lassen sich durch eine Rekursionsformel beschreiben:

$$\begin{aligned} d_c(C_F, C_r) \quad := \quad & \alpha_{i^*} d_c(C_{i^*}, C_r) + \alpha_{j^*} d_c(C_{j^*}, C_r) + \\ & \beta d_c(C_{i^*}, C_{j^*}) + \gamma |d_c(C_{i^*}, C_r) - d_c(C_{j^*}, C_r)| \end{aligned}$$

Parameter α_{i^*}, α_{j^*}, β und γ charakterisieren das jeweilige Clusterverfahren vollständig.

Varianten:

Verfahren	α_{i^*}	α_{j^*}	β	γ
single-linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
complete-linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
unweighted average	$\frac{1}{2}$	$\frac{1}{2}$	0	0
Median	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0
group average	$\frac{m_{i^*}}{m_{i^*}+m_{j^*}}$	$\frac{m_{j^*}}{m_{i^*}+m_{j^*}}$	0	0
centroid	$\frac{m_{i^*}}{m_{i^*}+m_{j^*}}$	$\frac{m_{j^*}}{m_{i^*}+m_{j^*}}$	$-\frac{m_{i^*} m_{j^*}}{(m_{i^*}+m_{j^*})^2}$	0
Ward's Verfahren	$\frac{m_{i^*}+m_r}{m_{i^*}+m_{j^*}+m_r}$	$\frac{m_{j^*}+m_r}{m_{i^*}+m_{j^*}+m_r}$	$-\frac{m_r}{m_{i^*}+m_{j^*}+m_r}$	0

m_k bezeichnet hierbei die Anzahl der Objekte im Cluster C_k.

Single-Linkage-Verfahren

Es ist $\alpha_{i^*} = \alpha_{j^*} = 1/2$, $\beta = 0$ und $\gamma = -1/2$, also

$$d_c(C_F, C_r) = \frac{1}{2} d_c(C_{i^*}, C_r) + \frac{1}{2} d_c(C_{j^*}, C_r) - \frac{1}{2} |d_c(C_{i^*}, C_r) - d_c(C_{j^*}, C_r)|$$

Wegen $\frac{x+y}{2} - \frac{|x-y|}{2} = \min\{x, y\}$ erhalten wir:

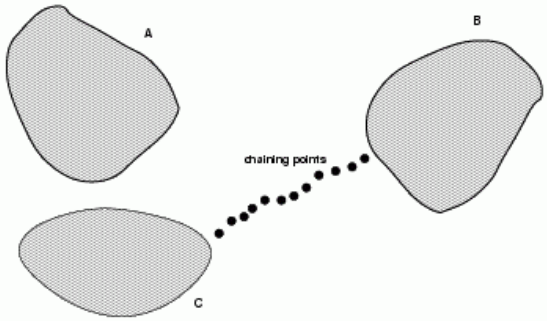
$$d_c(C_F, C_r) = \min\{d_c(C_{i^*}, C_r), d_c(C_{j^*}, C_r)\}$$

- Auf jeder Fusionsstufe werden die beiden Cluster vereinigt, die die zueinander am nächsten liegenden Nachbarobjekte, den *nearest neighbours*, besitzen.
- Deshalb bekannt auch als *nearest neighbour* Clusterverfahren (Vorsicht! nicht mit K-nearst-neighbour-Klassifikator verwechseln).
- Single-Linkage gehört zu den ältesten und einfachsten Clusteranalyseverfahren (Sneath 1957).

Chaining-Effekt

Nach dem Single-Linkage-Verfahren werden in der Abbildung in den nächsten Fusionsschritten nicht die Cluster **A** und **C** fusioniert, sondern die visuell separierten Cluster **C** und **B**, da diese durch sogenannte *chaining points* verbunden sind.

Chaining führt dazu, dass Distanzen zwischen Objekte eines Clusters häufig größer sind, als Distanzen zwischen Objekten verschiedener Cluster.



Complete-Linkage-Verfahren

Es ist $\alpha_{i^*} = \alpha_{j^*} = 1/2$, $\beta = 0$ und $\gamma = 1/2$.

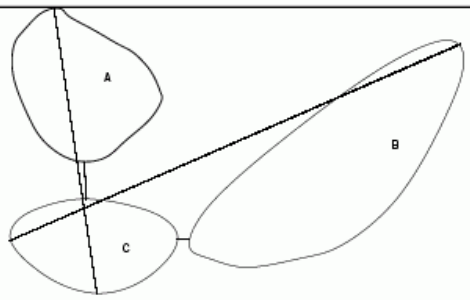
$$d_c(C_F, C_r) = \frac{1}{2} d_c(C_{i^*}, C_r) + \frac{1}{2} d_c(C_{j^*}, C_r) + \frac{1}{2} |d_c(C_{i^*}, C_r) - d_c(C_{j^*}, C_r)|$$

Wegen $\frac{x+y}{2} + \frac{|x-y|}{2} = \max\{x, y\}$ erhalten wir:

$$d_c(C_F, C_r) = \max\{d_c(C_{i^*}, C_r), d_c(C_{j^*}, C_r)\}$$

- Auf jeder Fusionsstufe werden die beiden Cluster fusioniert, die über die minimale Maximaldistanz von zwei Objekten verfügen (*minimal furthest neighbours*)
- Ebenfalls unter dem Namen *furthest neighbours* Clusterverfahren bekannt (Mac Naughton-Smith 1965)
- Complete-Linkage zählt ebenfalls zu den ältesten und unkompliziertesten Clusteranalyseverfahren (Soerensen 1948)

Complete- vs. Single-Linkage

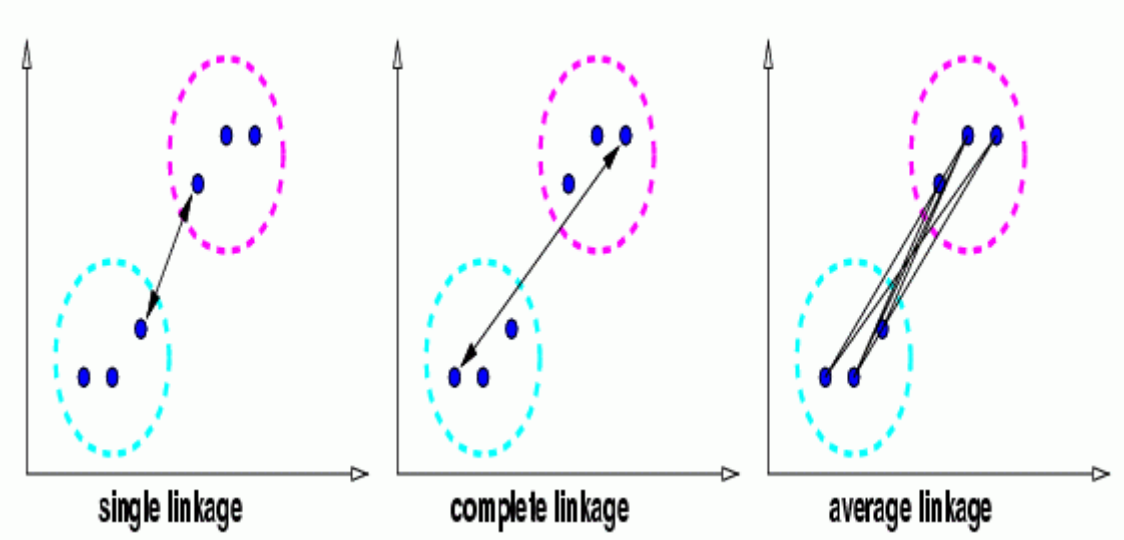


Fusion

- Single-Linkage: Cluster **C** und Cluster **B**
- Complete-Linkage: Cluster **C** und Cluster **A**

- Beim Complete-Linkage werden zwei Cluster nicht auf der Basis einer einzelnen kleinen Distanz zwischen Objektpaaren (*single link*) fusioniert, sondern die Distanzen aller Objektpaare werden betrachtet (*complete link*).
- Chaining tritt beim Complete-Linkage nicht auf.
- Complete-Linkage und Single-Linkage sind gewissermaßen die beiden Extreme bei der Verrechnung der Fusionsdistanzen; die anderen agglomerativen Clusterverfahren sind dazwischen einzuordnen.

(Schwenker 2004)



single-linkage

$$d_{\mathsf{SL}}(A,B) \stackrel{\text{def}}{=} \min_{\boldsymbol{x}\in A} \min_{\boldsymbol{y}\in B} d(\boldsymbol{x},\boldsymbol{y})$$

complete-linkage

$$d_{\mathsf{CL}}(A,B) \stackrel{\text{def}}{=} \max_{\boldsymbol{x}\in A} \max_{\boldsymbol{y}\in B} d(\boldsymbol{x},\boldsymbol{y})$$

average-linkage

$$d_{\mathsf{AL}}(A,B) \stackrel{\text{def}}{=} \frac{1}{|A|\cdot|B|} \sum_{\boldsymbol{x}\in A} \sum_{\boldsymbol{y}\in B} d(\boldsymbol{x},\boldsymbol{y})$$

(Schukat-Talamazzini 2002)

Verwendung der "average-linkage"-Mengendistanz:

Group-Average-Verfahren

$\alpha_{i^*} = \frac{m_{i^*}}{m_{i^*}+m_{j^*}}$, $\alpha_{j^*} = \frac{m_{j^*}}{m_{i^*}+m_{j^*}}$ und $\beta = \gamma = 0$.

Dann hat die Rekursionsformel die folgende Form:

$$d_c(C_F, C_r) = \frac{1}{m_{i^*} + m_{j^*}} \Big(m_{i^*} d_c(C_{i^*}, C_r) + m_{j^*} d_c(C_{j^*}, C_r) \Big)$$

Die Formel lässt sich in die iterative Form bringen:

$$d_c(C_F, C_r) = \frac{1}{(m_{i^*} + m_{j^*}) m_r} \sum_{e_i \in C_F} \sum_{e_j \in C_r} d(e_i, e_j)$$

Die Distanz zwischen den beiden Cluster C_F und C_r wird durch das arithmetische Mittel der Distanzen aller Objektpaare der beiden beteiligten Cluster C_F und C_r definiert.

Bei einem Fusionschritt werden somit die beiden Cluster fusioniert, für die das arithmetische Mittel aller Objektdistanzen minimal ist.

Centroid-Verfahren

$\alpha_{i^*} = \frac{m_{i^*}}{m_{i^*}+m_{j^*}}$, $\alpha_{j^*} = \frac{m_{j^*}}{m_{i^*}+m_{j^*}}$ und ferner sind $\beta = -\frac{m_{i^*} m_{j^*}}{m_{i^*}+m_{j^*}}$ und $\gamma = 0$.

Dann hat die Rekursionsformel die folgende Form (mit $m_F := m_{i^*} + m_{j^*}$):

$$d_c(C_F, C_r) = \frac{m_{i^*}}{m_F} d_c(C_{i^*}, C_r) + \frac{m_{j^*}}{m_F} d_c(C_{j^*}, C_r) - \frac{m_{i^*} m_{j^*}}{m_F^2} d_c(C_{i^*}, C_{j^*})$$

Idee beim Centroid-Verfahren:

- Euklidische Metrik als Abstandsfunktion.
- Objekte e_μ sind durch Vektoren $x_\mu \in \mathbb{R}^n$ beschrieben.
- Cluster C_i sind repräsentiert durch **Prototypen/Clusterzentren** $c_i \in \mathbb{R}^n$.
- Clusterzentren c_i liegen im Schwerpunkt der Datenvektoren x_μ, die den Objekten des Clusters C_i zugeordnet sind.

weitere Varianten: Median-Verfahren, Unweighted-average-Verfahren, Ward's Verfahren (s. Schwenker 2004)

eine hierarchische Clusterung lässt sich auch während der Konstruktion eines *minimalen aufspannenden Baumes* erzeugen:

Minimal Spanning Tree

- Graph $G = (V, E)$, V Knotenmenge und $E \subset V \times V$ Kantenmenge.
- G heisst zusammenhängend, falls für alle $v_i, v_j \in V$ ein Pfad $(v_i, \ldots, v_j) \in E^l$ $l \geq 2$ existiert. Pfad $(v_i, \ldots, v_j)$ heisst geschlossen, falls $v_i = v_j$ gilt.
- Ein Baum ist ein zusammenhängender Graph ohne geschlossene Pfade.
- $G(V, E)$ ein Baum, dann ist $|E| = |V| - 1$.
- Ein (auf-)spannender Baum eines Graphen, ist ein Baum, der sämtliche Knoten enthält.
- Sind die Kanten eines Graphen gewichtet, so ist ein minimaler spannender Baum (minimal spanning tree), ein Baum mit minimalem Kantensumme, also Summe aller Kantengewichte des spannenden Baumes

MST-Algorithmus

- Input: Graph $G = (V, E)$, Kantengewichte $w : E \to \mathbb{R}$ mit $w(v_i, v_j) = w_{ij}$
- Output: $T_{\min} = (V_T, E_T)$ mit $V_T = V$ und mit

$$\sum_{(v_i,v_j)\in E_T} w_{ij} = \min_{T\subset G, T\text{Baum}} \sum_{(v_i,v_j)\in T} w_{ij}$$

1. Wähle $v_i \in V$ und setze $V_T = \{v_i\}$ und $E_T = \emptyset$.
2. Bestimme $v_{i^*} \in V_T$ und $v_{j^*} \in V \setminus V_T$ mit $(v_{i^*}, v_{j^*}) \in E$ und mit

   $$w_{i^*j^*} = \min\{w_{ij} \ : \ v_i \in V_T,\ j \in V \setminus V_T,\ (v_i, V_j) \in E\}$$

   Setze dann $V_T = V_T \cup \{v_{j^*}\}$ und $E_T = E_T \cup (v_{i^*}, v_{j^*})$.
3. Goto 2. bis $V_T = V$

(Schwenker 2004)

Hierarchische Verfahren (Single-Link und Varianten):

Diskussion

+ erfordert keine Kenntnis der Anzahl k der Cluster
+ findet nicht nur ein flaches Clustering, sondern eine ganze Hierarchie
+ ein einzelnes Clustering kann aus dem Dendrogramm gewonnen werden, z.B. mit Hilfe eines horizontalen Schnitts durch das Dendrogramm
  (erfordert aber wieder Anwendungswissen)

- Entscheidungen können nicht zurückgenommen werden
- Anfälligkeit gegenüber Rauschen (Single-Link)
  eine „Linie“ von Objekten kann zwei Cluster verbinden
- Ineffizienz
  Laufzeitkomplexität von mindestens $O(n^2)$ für n Objekte

weitere Variante:

CURE [Guha, Rastogi & Shim 1998]

• Motivation

(a) (b)

• Repräsentation eines Clusters
  partitionierende Verfahren: ein Punkt
  hierarchische Verfahren: alle Punkte
• CURE: Repräsentation eines Clusters durch c Repräsentanten
• Entdecken nicht-konvexer Cluster
• Vermeidung des Single-Link Effekts

Unterschiede zu Single Link

- Jedem Cluster, der genügend groß ist, werden c Repräsentanten zugeordnet
- Diese werden so gewählt, dass sie im Cluster gleichmäßig verteilt sind
- Dann werden die Punkte um einen Faktor α zum Zentroiden hin verschoben

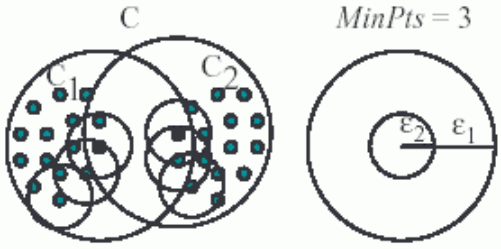
Folge: Anpassung an Form des Clusters ohne Single-Link-Effekt

- Außerdem arbeitet das hierarchische Verfahren nur auf einem Sample
- Nachträgliche Zuordnung der DB-Punkte zu den Clustern

Dichtebasiertes hierarchisches Clustering

Grundlagen [Ankerst, Breunig, Kriegel & Sander 1999]

- für einen konstanten *MinPts*-Wert sind dichte-basierte Cluster bzgl. eines kleineren ε vollständig in Clustern bzgl. eines größeren ε enthalten



- in einem DBSCAN-ähnlichen Durchlauf gleichzeitig das Clustering für verschiedene Dichte-Parameter bestimmen

zuerst die dichteren Teil-Cluster, dann den dünneren Rest-Cluster

- kein Dendrogramm, sondern eine auch noch bei sehr großen Datenmengen übersichtliche Darstellung der Cluster-Hierarchie

Grundbegriffe

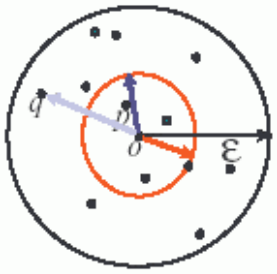
Kerndistanz eines Objekts p bzgl. ε und *MinPts*

$$Kerndistanz_{\varepsilon,MinPts}(o) = \begin{cases} UNDEFINIERT, \text{ wenn } |N_\varepsilon(o)| < MinPts \\ MinPtsDistanz(o), \text{ sonst} \end{cases}$$

Erreichbarkeitsdistanz eines Objekts p relativ zu einem Objekt o

$$Erreichbarkeitsdistanz_{\varepsilon,MinPts}(p,o) = \begin{cases} UNDEFINIERT, \text{ wenn } |N_\varepsilon(o)| < MinPts \\ \max\{Kerndistanz(o), dist(o,p)\}, \text{ sonst} \end{cases}$$

$MinPts = 5$



Kerndistanz(o)

Erreichbarkeitsdistanz(p,o)

Erreichbarkeitsdistanz(q,o)

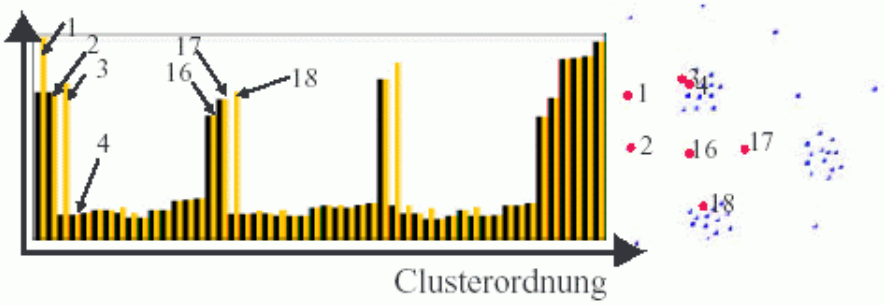
→ Algorithmus OPTICS (s. Ester & Sander 2000, S. 81)

Clusterordnung

- OPTICS liefert nicht direkt ein (hierarchisches) Clustering, sondern eine „Clusterordnung" bzgl. ε und *MinPts*
- *Clusterordnung* bzgl. ε und *MinPts*
  - beginnt mit einem beliebigen Objekt
  - als nächstes wird das Objekt besucht, das zur Menge der bisher besuchten Objekte die minimale Erreichbarkeitsdistanz besitzt

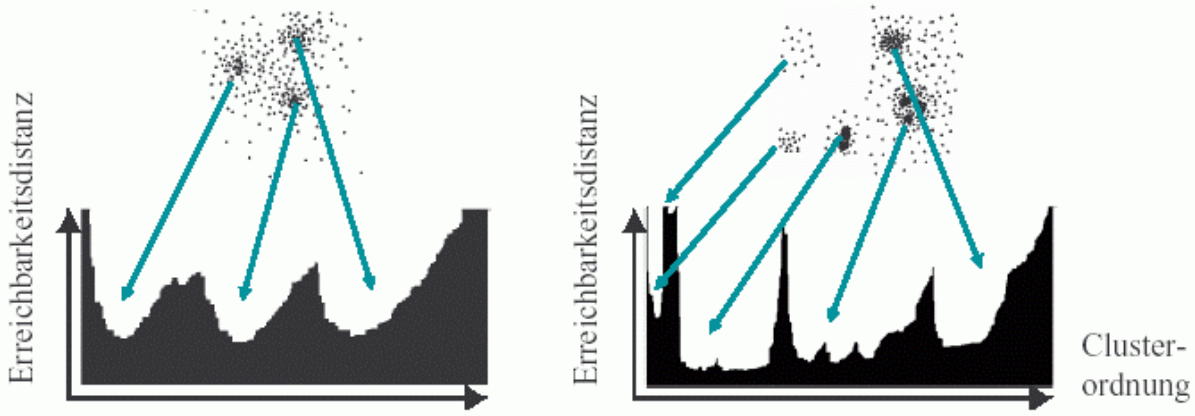
Kerndistanz

Erreichbarkeits-distanz

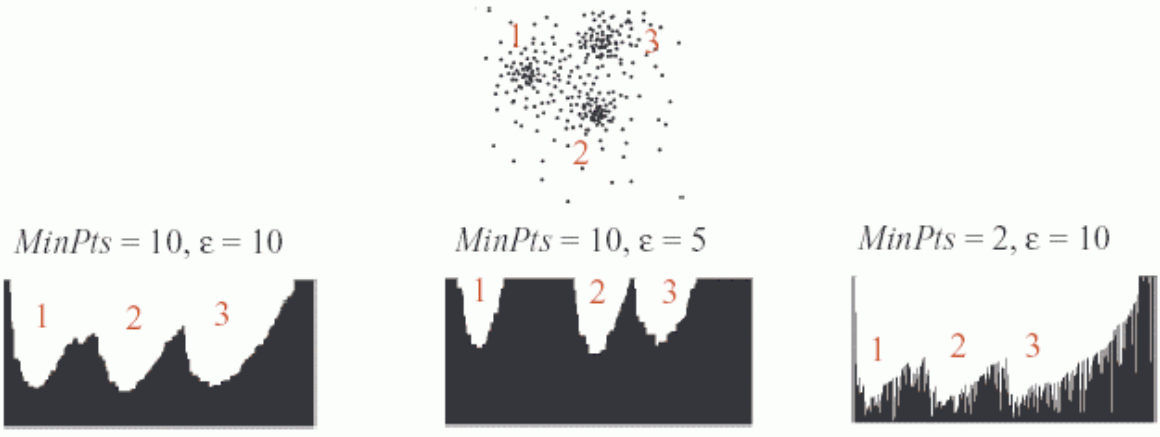


Erreichbarkeits-Diagramm

- Zeigt die Erreichbarkeitsdistanzen (bzgl. ε und *MinPts*) der Objekte als senkrechte, nebeneinanderliegende Balken
- in der durch die Clusterordnung der Objekte gegebenen Reihenfolge



Parameter-Sensitivität



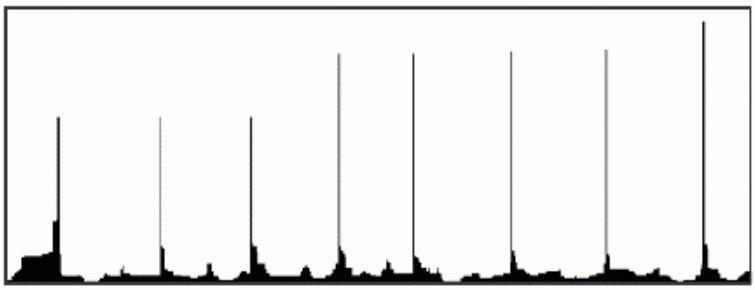
optimale Parameter | kleineres ε | kleineres *MinPts*

Clusterordnung ist robust gegenüber den Parameterwerten
gute Resultate wenn Parameterwerte „groß genug“

Manuelle Analyse der Cluster

Mit Erreichbarkeits-Diagramm

- gibt es Cluster?
- wieviele Cluster?
- sind die Cluster hierarchisch geschachtelt?
- wie groß sind die Cluster?

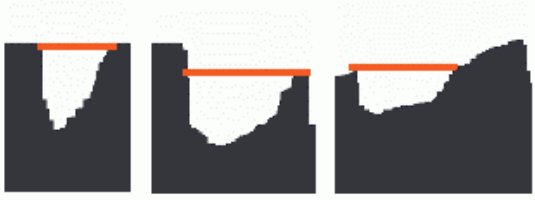


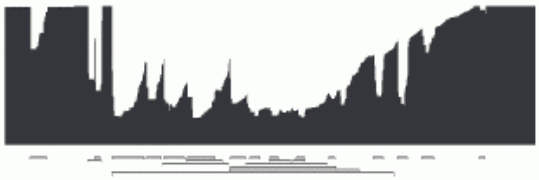
Erreichbarkeits-Diagramm

Automatisches Entdecken von Clustern

ξ-Cluster

- Teilsequenz der Clusterordnung
- beginnt in einem Gebiet ξ-steil *abfallender* Erreichbarkeitsdistanzen
- endet in einem Gebiet ξ-steil *steigender* Erreichbarkeitsdistanzen bei etwa demselben absoluten Wert
- enthält mindestens *MinPts* Punkte





Algorithmus

- bestimmt alle ξ-Cluster
- markiert die gefundenen Cluster im Erreichbarkeits-Diagramm
- Laufzeitaufwand O(*n*)

(Böhm 2003)

Entdeckung von Ausreißern (Outlier detection)

Was ist ein "Ausreißer"?
- keine allgemein akzeptierte Definition

"one person's noise could be another person's signal"

Anwendungen:

- Kreditkarten-Mißbrauch
- Telefonkunden-Betrug
- Medizinische Analyse

Definitionen:

- Nach Hawkins (1980) : "Ein Outlier ist eine *Beobachtung*, die sich von den anderen *Beobachtungen* so deutlich unterscheidet, daß man denken könnte, sie sei von einem anderen Mechanismus generiert worden."

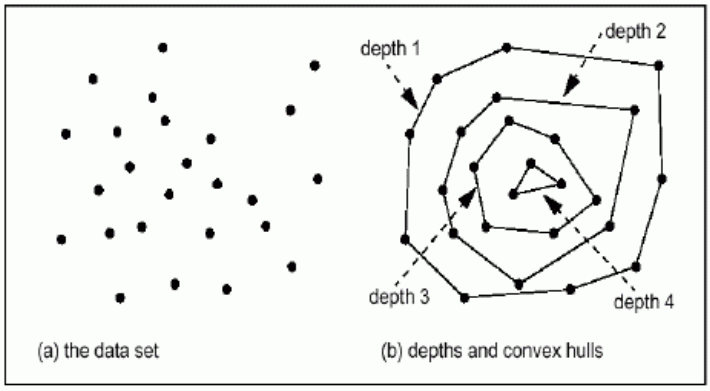
Identifikation von Outlier

Depth-based :

- Jedes Objekt wird als ein Punk in einem d-Dimensionalen Raum betrachtet.
- Die Tiefe der Objekte wird berechnet.
- Outlier haben eine kleinere Tiefe als die anderen Objekte.

Theoretisch :
=> gut auch für hoch.dim. Daten

Praktisch :
=> ineffizient für $d \geq 4$,
da die konvexe Hülle berechnet werden muss.



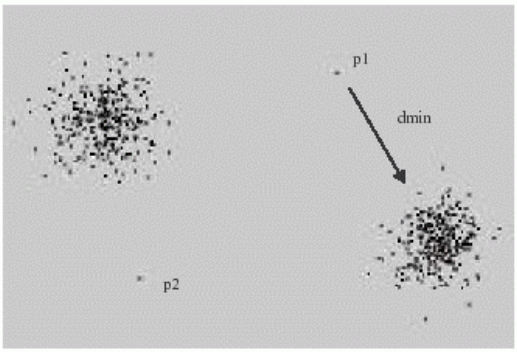
Distance-based : [Knorr, Ng 97]

- Ein Objekt p in einem Datensatz D ist ein *DB(pct,dmin)*-Outlier (DB = distance-based), falls mindestens *pct* - Prozent von Objekten aus D eine größere Distanz als *dmin* zu p haben.

  - Wahl von *pct* und *dmin* wird einem Experten überlassen.

Beispiel: $p_1 \in D$, $pct=0.95$, $dmin=8$

DB(0.95,8) =>
95% von Objekten aus D haben eine Distanz > *8* zu p_1



jedoch: diese Definition ist für viele Zwecke zu unflexibel

- Nicht nur binäre Eigenschaften für Outlier (Outlier? JA oder NEIN)
- Bei Clustern mit unterschiedlicher Dichte, können beim *distance-based* - Ansatz Probleme auftreten (*DB(pct,dmin)*)
  - „Globale“ Outlier vs. „lokale“ Outlier

Lösung : *Density-based Local Outlier* : [Breunig, Kriegel, Ng, Sander 2000]

- Weise jedem Objekt einen *Grad* zu, zu dem das Objekt ein Outlier ist
  - Local Outlier Factor (LOF)
- Lokale Nachbarschaft von Objekten wird berücksichtigt

Grundlagen

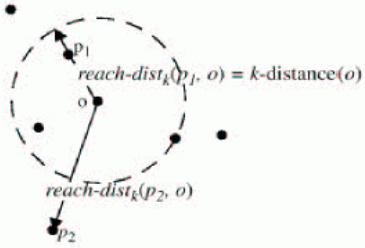
- *k-Distanz* von p = *dist(p,o)*, für jedes k, so dass gilt: ($o \in D$)
  - (i) für mindestens k Objekte $q \in D$ gilt : $dist(p,q) \leq dist(p,o)$
  - (ii) für höchstens $k-1$ Objekte $q \in D$ gilt : $dist(p,q) < dist(p,o)$

- *k-Distanz - Nachbarschaft* von p:

$$N_{k\text{-distance}(p)}(p) = \{q \in D \setminus \{p\} \mid dist(p,q) \leq k\text{-distance}(p)\}$$

- *Erreichbarkeits-Distanz* :

$$reach\text{-}dist_k(p,o) = \max\{k\text{-distance}(o), dist(p,o)\}$$



- Als Parameter nur *MinPts*
- Lokale Erreichbarkeits-Distanz von p:

$$lrd_{MinPts}(p) = 1 / \left(\frac{\sum_{o \in N_{MinPts}(p)} reach\text{-}dist_{MinPts}(p, o)}{|N_{MinPts}(p)|} \right)$$

- Local Outlier Factor von p (LOF):

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

- LOF $(p) \approx 1$: Punkt liegt weit innen im Cluster
- LOF $(p) >> 1$: Punkt ist ein starker lokaler Outlier

Anwendungen:

Beispiele

Szenario:

- Analysiere die Bewegung eines Objektes (z.B. des Menschen) von Punkt A zu Punkt B.
- Alarmiere Benutzer, falls sich das Objekt nicht an vorgeschriebene Routen hält.

Andere Beispiele:

- Erdbeben früh erkennen (Quakefinder)
  - durch Satellitenbeobachtungen
- Vulkane auf Venus erkennen (JARtool)

[Knorr, Ng, Tucakov 2000]: Triclops

(Böhm 2003)

wegen Verwendung von Abständen in dieses Kapitel aufgenommen, aber kein Clustering-, sondern Klassifikations-Verfahren:

Nächste-Nachbarn-Klassifikatoren

hier wieder Datensatz mit Klassen vorgegeben

Beispiel:

- Instanzbasiertes Lernen (*instance based learning*)
- Einfachster Nächste-Nachbar-Klassifikator:
  Zuordnung zu der Klasse des nächsten Nachbarpunkts
- Im Beispiel: Nächster Nachbar ist eine Schraube
- Regionen der Klassenzuordnung können als *Voronoi-Diagramme* dargestellt werden:

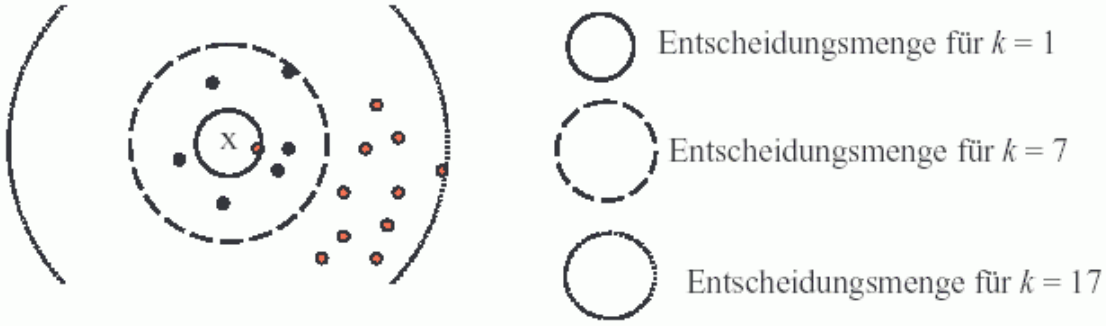
Fehlzuordnung durch Ausreißer möglich ⇒

- Besser: Betrachte mehr als nur einen Nachbarn
  k-Nächste-Nachbarn-Klassifikator
- *Entscheidungsmenge*
  die Menge der zur Klassifikation betrachteten *k*-nächsten Nachbarn
- *Entscheidungsregel*
  wie bestimmt man aus den Klassen der Entscheidungsmenge die Klasse des zu klassifizierenden Objekts?
  - Interpretiere Häufigkeit einer Klasse in der Entscheidungsmenge als Wahrscheinlichkeit der Klassenzugehörigkeit
  - Maximum-Likelihood-Prinzip: Mehrheitsentscheidung
  - Ggf. Gewichtung

"kNN-Klassifikator"

Wahl des Parameters k

- „zu kleines“ k: hohe Sensitivität gegenüber Ausreißern
- „zu großes“ k: viele Objekte aus anderen Clustern (Klassen) in der Entscheidungsmenge.
- mittleres k: höchste Klassifikationsgüte, oft $1 << k < 10$



x: zu klassifizieren

Entscheidungsregel

Standardregel

wähle die Mehrheitsklasse der Entscheidungsmenge

Gewichtete Entscheidungsregel

gewichte die Klassen der Entscheidungsmenge

- nach Distanz, meist invers quadriert: $weight\ (dist) = 1/dist^2$
- nach Verteilung der Klassen (oft sehr ungleich!)
  Problem: Klasse mit zu wenig Instanzen ($< k/2$) in der Trainingsmenge bekommt keine Chance, ausgewählt zu werden, selbst bei optimaler Distanzfunktion
  - Klasse A: 95 %, Klasse B 5 %
  - Entscheidungsmenge = {A, A, A, A, B, B, B}
  - Standardregel $\Rightarrow$ A, gewichtete Regel $\Rightarrow$ B

(Böhm 2003)

Zusammenfassung zum kNN-Klassifikator:

Der k-**Nearest-Neighbour-Klassifikator** berechnet zur Klassifikation eines Objektes die k ähnlichsten Objekte aus dem Datensatz und ordnet das gegebene Objekt in die Klasse, die am häufigsten unter den k ähnlichsten Objekten vorkommt.

Die Ähnlichkeit bzw. die Unähnlichkeit zweier Objekte wird meistens über den euklidischen Abstand definiert. Dafür sollten die Attribute vorher normalisiert oder standardisiert werden.

k wird üblicherweise durch Probieren ermittelt.

Diskussion:

- k-Nearest-Neighbour-Klassifikatoren eignen sich vor allem für Datensätze mit vorwiegend numerischen Attributen.
- k-Nearest-Neighbour-Klassifikatoren erfordern keine Lernphase/Training (außer evtl. zur Bestimmung von k). Bei sehr großen Datensätzen kann die Suche nach den k ähnlichsten Objekten sehr lange dauern. In diesem Fall wählt man eine (repräsentative) Stichprobe aus dem Datensatz aus oder wendet zunächst eine Clusteranalyse an.
- Bei Datensätzen mit sehr vielen Attributen ist die euklidische Distanz meistens wenig aussagekräftig, so dass Nearest-Neighbour-Klassifikatoren in diesem Fall meistens versagen.

(Klawonn 2004)