# 3. Clustering

*Inhalt dieses Kapitels*

## 3.1 Einleitung

Ziel des Clustering, Distanzfunktionen, Anwendungen,
Typen von Algorithmen

## 3.2 Partitionierende Verfahren

k-means, k-medoid, Expectation Maximization, Initialisierung und
Parameterwahl, Probleme optimierender Verfahren, dichtebasierte Verfahren

## 3.3 Hierarchische Verfahren

Single-Link und Varianten, dichtebasiertes hierarchisches Clustering

# 3. Clustering

*Inhalt dieses Kapitels*

3.4 Begriffliches Clustern

Formale Begriffsanalyse, Begriffsverbände

3.5 Datenbanktechniken zur Leistungssteigerung

Indexunterstütztes Sampling, Indexunterstützte Anfragebearbeitung, Datenkompression mit BIRCH

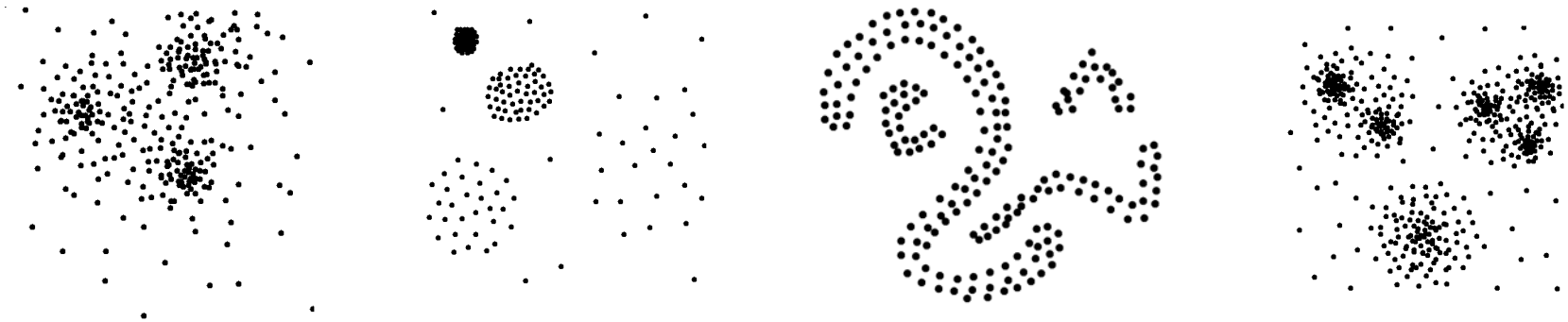
3.6 Besondere Anforderungen und Verfahren

k-modes, verallgemeinertes dichtebasiertes Clustering, inkrementelles Clustering, Subspace Clustering

# 3.1 Einleitung

## *Ziel des Clustering*

- Identifikation einer endlichen Menge von Kategorien, Klassen oder Gruppen (*Cluster*) in den Daten
- Objekte im *gleichen* Cluster sollen möglichst ähnlich sein
- Objekte aus *verschiedenen* Clustern sollen möglichst unähnlich zueinander sein



Cluster unterschiedlicher Größe, Form und Dichte

hierarchische Cluster

# 3.1 Distanzfunktionen

## *Grundbegriffe*

### Formalisierung der Ähnlichkeit

- manchmal: Ähnlichkeitsfunktion
- meist: Distanzfunktion $dist(o_1,o_2)$ für Paare von Objekten $o_1$ und $o_2$
- kleine Distanz ≈ ähnliche Objekte
- große Distanz ≈ unähnliche Objekte

### Anforderungen an Distanzfunktionen

(1) $dist(o_1, o_2) = d \in R^{\geq 0}$

(2) $dist(o_1, o_2) = 0$ genau dann wenn $o_1 = o_2$

(3) $dist(o_1, o_2) = dist(o_2, o_1)$ (Symmetrie)

(4) zusätzlich für Metriken (Dreiecksungleichung)

$$dist(o_1, o_3) \leq dist(o_1, o_2) + dist(o_2, o_3).$$

# 3.1 Distanzfunktionen

## *Distanzfunktionen für numerische Attribute*

Objekte $x = (x_1, ..., x_d)$ und $y = (y_1, ..., y_d)$

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

*Euklidische Distanz* ($p = 2$) $dist(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$

*Manhattan-Distanz* ($p = 1$) $dist(x, y) = \sum_{i=1}^{d} |x_i - y_i|$

*Maximums-Metrik* ($p = \infty$) $dist(x, y) = \max\{|x_i - y_i| | 1 \leq i \leq d\}$

eine populäre *Ähnlichkeitsfunktion*: Korrelationskoeffizient $\in$ [-1,+1]

# 3.1 Distanzfunktionen

## *Andere Distanzfunktionen*

- für kategorische Attribute (Hamming-Distanz) $dist(x,y) = \sum_{i=1}^{d} \delta(x_i, y_i)$ *mit* $\delta(x_i, y_i) = \begin{cases} 0 \text{ falls } x_i = y_i \\ 1 \text{ sonst} \end{cases}$
- für Textdokumente $D$ (Vektoren der Häufigkeit der Terme aus $T$)

$$d = \{g(f(t_i, D)) \mid t_i \in T\}$$

$f(t_i, D)$: Häufigkeit des Terms $t_i$ in Dokument $D$

$g$: monotone *Dämpfungsfunktion (z.B. Mult. mit inverser Dok.-Häufigkeit)*

$$dist(x,y) = 1 - \frac{< x, y >}{|x| \cdot |y|} \quad mit \; <.,.> Skalarprodukt \; und \; |.| \; Laenge \; des \; Vektors$$

Adäquatheit der Distanzfunktion ist wichtig für Qualität des Clustering

# 3.1 Typische Anwendungen

## *Überblick*

- Kundensegmentierung

  Clustering der Kundentransaktionen

- Bestimmung von Benutzergruppen auf dem Web

  Clustering der Web-Logs

- Strukturierung von großen Mengen von Textdokumenten

  Hierarchisches Clustering der Textdokumente

- Erstellung von thematischen Karten aus Satellitenbildern

  Clustering der aus den Rasterbildern gewonnenen Featurevektoren

# 3.1 Typische Anwendungen

## *Bestimmung von Benutzergruppen auf dem Web*

Einträge eines Web-Logs

```
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:44:50 +0100] "GET /~lopa/ HTTP/1.0" 200 1364
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:45:11 +0100] "GET /~lopa/x/ HTTP/1.0" 200 712
fixer.sega.co.jp unknown - [04/Mar/1997:01:58:49 +0100] "GET /dbs/porada.html HTTP/1.0" 200 1229
scooter.pa-x.dec.com unknown - [04/Mar/1997:02:08:23 +0100] "GET /dbs/kriegel_e.html HTTP/1.0" 200 1241
```

Generierung von Sessions

Session::= <IP-Adresse, Benutzer-Id, [$URL_1$, . . ., $URL_k$]>

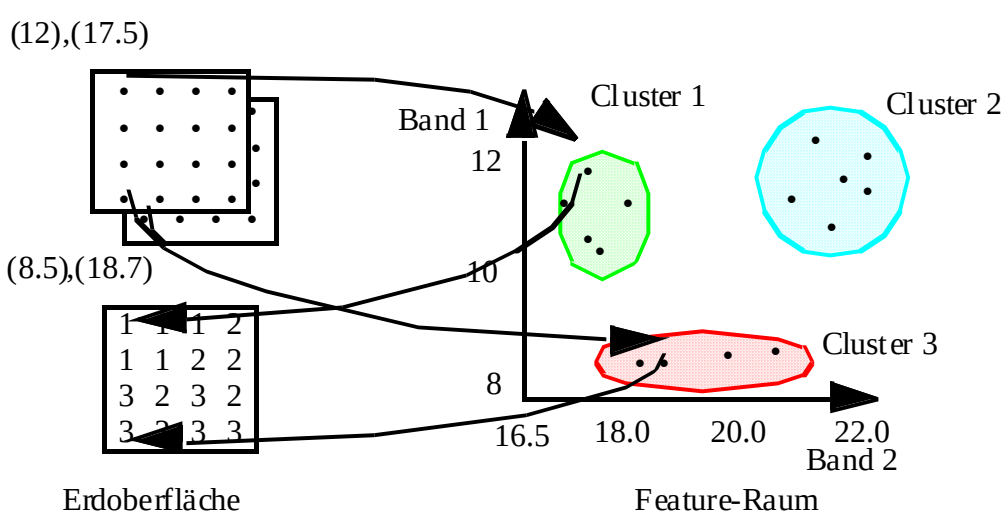
⇒ welche Einträge bilden eine Session?

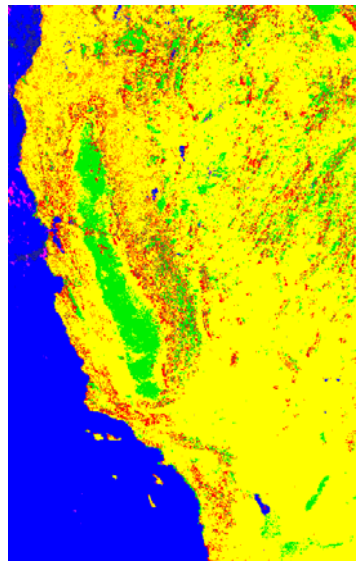
Distanzfunktion für Sessions

$$d(x,y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|}$$

# 3.1 Typische Anwendungen

*Erstellung von thematischen Karten aus Satellitenbildern*



Grundlage

verschiedene Oberflächenbeschaffenheiten der Erde besitzen jeweils ein charakteristisches Reflexions- und Emissionsverhalten

# 3.1 Typen von 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

# 3.2 Partitionierende Verfahren

## *Grundlagen*

### Ziel

eine Partitionierung in *k* Cluster mit minimalen Kosten

### Lokal optimierendes Verfahren

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

### Typen von Cluster-Repräsentanten

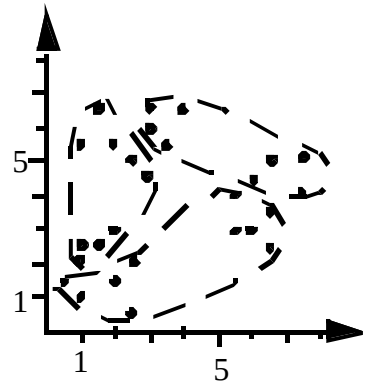
- Mittelwert des Clusters (*Konstruktion zentraler Punkte*)
- Element des Clusters (*Auswahl repräsentativer Punkte*)
- Wahrscheinlichkeitsverteilung des Clusters (*Erwartungsmaximierung*)

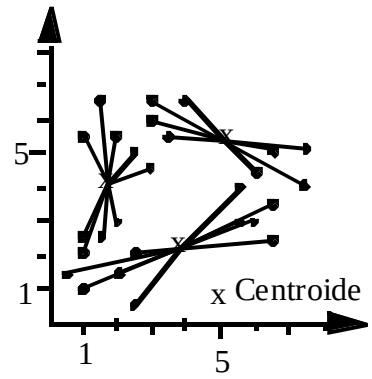
# 3.2 Konstruktion zentraler Punkte

*Beispiel*

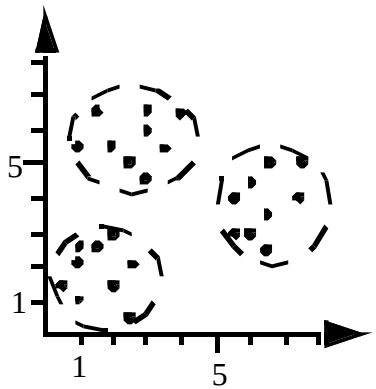
Cluster | Cluster-Repräsentanten

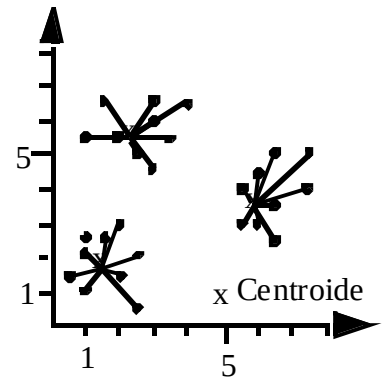
schlechtes Clustering




optimales Clustering



# 3.2 Konstruktion zentraler Punkte

## *Grundbegriffe* [Forgy 1965]

- Objekte sind Punkte $p=(x^p_1, ..., x^p_d)$ in einem euklidischen Vektorraum
- euklidische Distanz
- *Centroid* $\mu_C$: 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*

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

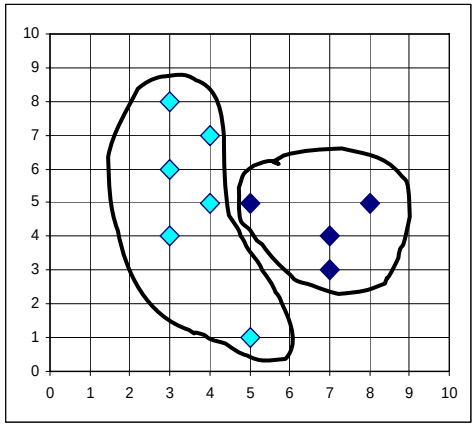
# 3.2 Konstruktion zentraler Punkte

*Algorithmus*

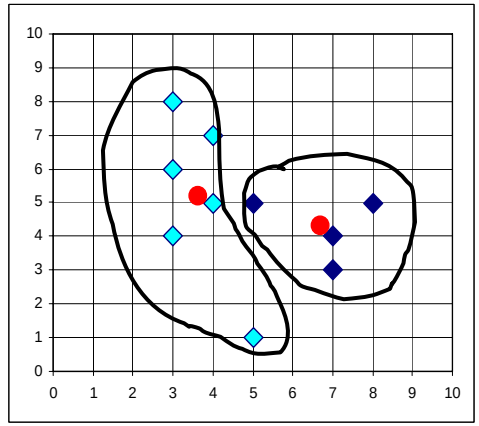
```
ClusteringDurchVarianzMinimierung(Punktmenge D,
Integer k)
  Erzeuge eine „initiale“ Zerlegung der Punktmenge D
   in k Klassen;
  Berechne die Menge C’={C1, ..., Ck} der Centroide
   für die k Klassen;
  C = {};
  repeat until C = C’
     C = C’;
     Bilde k Klassen durch Zuordnung jedes Punktes
      zum nächstliegenden Centroid aus C;
     Berechne die Menge C’={C’1, ..., C’k} der
      Centroide für die neu bestimmten Klassen;
  return C;
```

# 3.2 Konstruktion zentraler Punkte

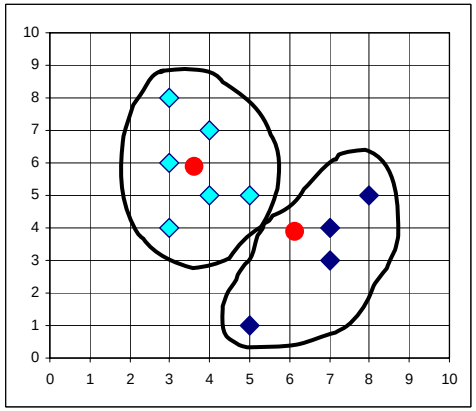
*Beispiel*



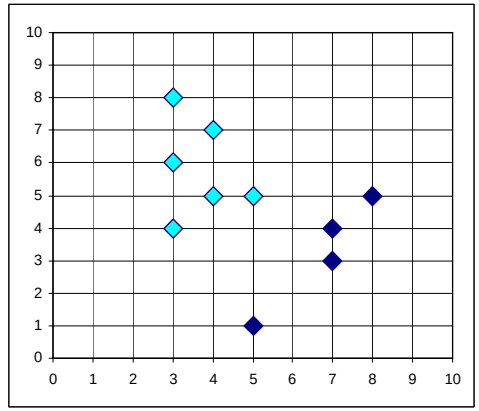

Berechnung der neuen Centroide




Zuordnung zum nächsten Centroid




Berechnung der neuen Centroide



# 3.2 Konstruktion zentraler Punkte

## *Varianten des Basis-Algorithmus*

*k-means* [MacQueen 67]

- Idee: die betroffenen Centroide werden direkt aktualisiert, wenn ein Punkt seine Clusterzugehörigkeit ändert
- *K*-means hat im wesentlichen die Eigenschaften des Basis-Algorithmus
- *K*-means 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

# 3.2 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 Centroids

- 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

# 3.2 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*

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

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

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

- Suchraum für den Clustering-Algorithmus: alle $k$-elementigen Partitionen der Datenbank $D$ mit $|D| = n$

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

# 3.2 Auswahl repräsentativer Punkte

## *Überblick über die 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

# 3.2 Auswahl repräsentativer Punkte

*Algorithmus PAM*

```
PAM(Objektmenge D, Integer k, Float dist)
  Initialisiere die k Medoide;
  TD_Änderung := −∞;
  while TD_Änderung < 0 do
     Berechne für jedes Paar (Medoid M, Nicht-Medoid N)
       den Wert TD_{N↔M};
     Wähle das Paar (M, N), für das der Wert
       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;
```

# 3.2 Auswahl repräsentativer Punkte

*Algorithmus CLARANS*

```
CLARANS(Objektmenge D, Integer k, Real dist,
        Integer numlocal, Integer maxneighbor)
  for r from 1 to numlocal do
    wähle zufällig k Objekte als Medoide; i := 0;
    while i < maxneighbor do
         Wähle zufällig (Medoid M, Nicht-Medoid N);
         Berechne TD_Änderung := TD_{N↔M} – TD;
         if TD_Änderung < 0 then
           ersetze M durch N;
           TD := TD_{N↔M}; i := 0;
         else i:= i + 1;
    if TD < TD_best then
         TD_best := TD; Merke die aktuellen Medoide;
  return Medoide;
```

# 3.2 Auswahl repräsentativer Punkte

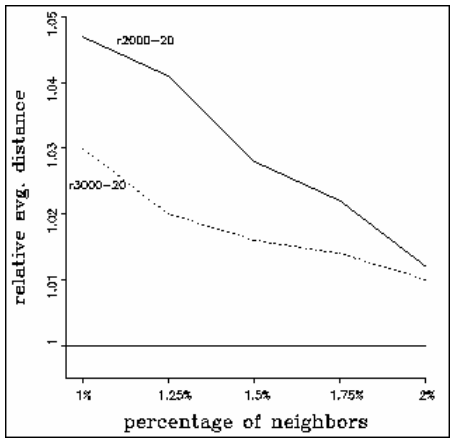
## *Vergleich von PAM und CLARANS*

Laufzeitkomplexitäten

- PAM: O($n^3 + k(n-k)^2 * \#Iterationen$)
- CLARANS O(*numlocal* * *maxneighbor* * #Ersetzungen * *n*)
  praktisch O($n^2$)

Experimentelle Untersuchung

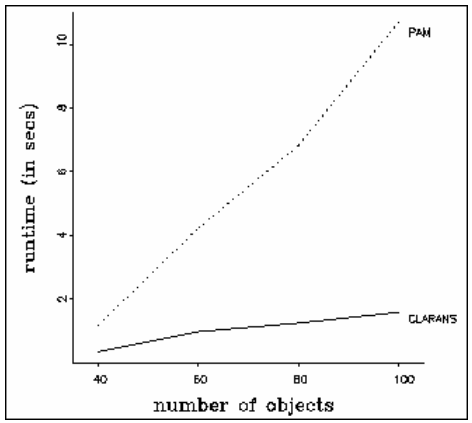
Qualität




TD(CLARANS)

TD(PAM)

Laufzeit



# 3.2 Erwartungsmaximierung

## *Grundbegriffe* [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
- typischerweise: Gaußverteilung (Normalverteilung)
- Repräsentation eines Clusters *C*
  - Mittelwert $\mu_C$ aller Punkte des Clusters
  - *d x d* Kovarianzmatrix $\Sigma_C$ für die Punkte im Cluster *C*

- Wahrscheinlichkeitsdichte eines Clusters *C*

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

# 3.2 Erwartungsmaximierung

## *Grundbegriffe*

- Wahrscheinlichkeitsdichte eines Clusterings $M = \{C_1, \ldots, C_k\}$

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

mit $W_i$ Anteil der Punkte aus $D$ in $C_i$

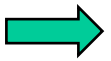
- Zuordnung von Punkten zu Clustern

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

Punkt gehört zu mehreren Clustern mit unterschiedlicher Wahrscheinlichkeit

- Maß für die Güte (Wahrscheinlichkeit) eines Clustering $M$

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



je größer der Wert $E$ ist, desto wahrscheinlicher sind die gegebenen Daten $D$ , geg. die berechnete Verteilung

$E(M)$ soll maximiert werden

# 3.2 Erwartungsmaximierung

*Algorithmus*

**ClusteringDurchErwartungsmaximierung**
(Punktmenge D, Integer k)
  Erzeuge ein „initiales“ Modell M’ = ($C_1$’, ..., $C_k$’);
  **repeat** // „Neuzuordnung“
    Berechne P(x|$C_i$), P(x) und P($C_i$|x) für jedes Objekt aus D und jede Gaußverteilung/jeden Cluster $C_i$;
    // „Neuberechnung des Modells“
    Berechne ein neues Modell M ={$C_1$, ..., $C_k$} durch Neuberechnung von $w_i$, $\mu_C$ und $\Sigma_C$ für jedes i;
    M’ := M;
  **until** |E(M) - E(M’)| < $\varepsilon$;
  **return** M;

# Neuberechnung der Parameter

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

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

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

# 3.2 Erwartungsmaximierung

## *Diskussion*

- Konvergiert gegen ein (möglicherweise nur *lokales*) Minimum
- Aufwand:

  O($n$ ∗ |M| * *#Iterationen*)

  Anzahl der benötigten Iterationen im allgemeinen sehr hoch

- Ergebnis und Laufzeit hängen 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.

# 3.2 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

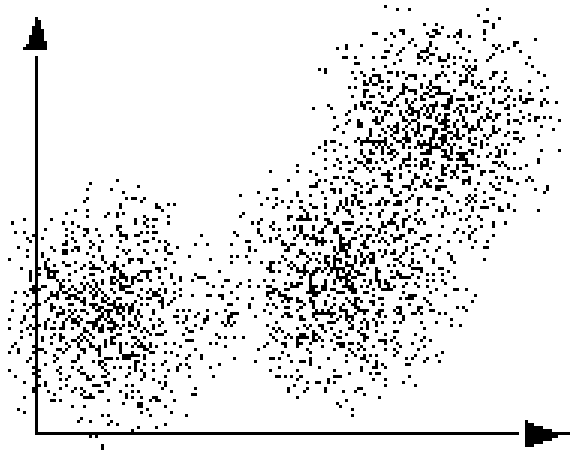
  ⇒ *m* verschiedene Schätzungen für *k* Clusterzentren

  $A = (A_1, A_2, \ldots, A_k)$, $B = (B_1, \ldots, B_k)$, $C = (C_1, \ldots, C_k)$, . . .
- Clustere nun die Menge $DB = A \cup B \cup C \cup \ldots$

  mit *m* verschiedenen Initialisierungen *A*, *B*, *C*, . . .
- Wähle von den *m* Clusterings dasjenige mit dem besten Wert bezüglich des zugehörigen Maßes für die Güte eines Clustering

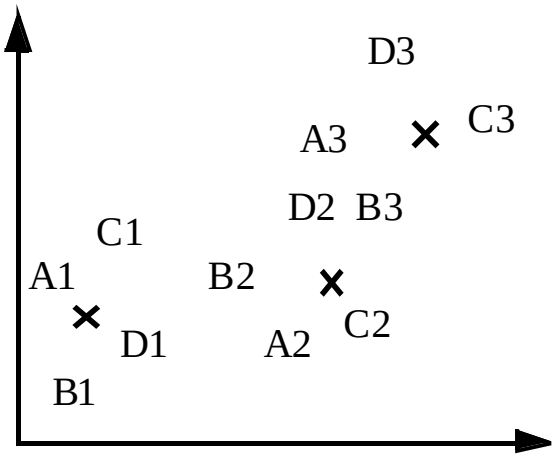
# 3.2 Wahl des initialen Clustering

*Beispiel*



Grundgesamtheit

$k = 3$ Gauß-Cluster




DB

von $m = 4$ Stichproben

× *wahre Clusterzentren*

# 3.2 Wahl des Parameters *k*

## Methode

- Bestimme für $k = 2, ..., n-1$ 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* sinkt monoton mit steigendem *k*

# 3.2 Wahl des Parameters *k*

## *Silhouetten-Koeffizient* [Kaufman & Rousseeuw 1990]

- ein von *k* unabhängiges Gütemaß für die *k*-means- und *k*-medoid-Verfahren
- 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 $o$

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

  $s(o)$ = -1 / 0 / +1: schlechte / indifferente / gute Zuordung

- Silhouettenkoeffizient $s_C$ eines Clustering

  durchschnittliche Silhouette aller Objekte

- Interpretation des Silhouettenkoeffizients

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

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

# 3.2 Dichtebasiertes Clustering

## *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

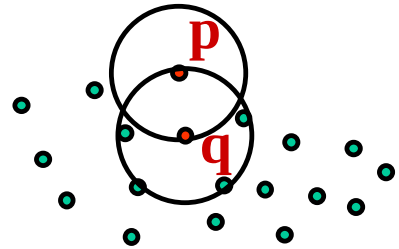
- für jedes Objekt eines Clusters überschreitet die lokale Punktdichte einen gegebenen Grenzwert
- die Menge von Objekten, die den Cluster ausmacht, ist räumlich zusammenhängend

# 3.2 Dichtebasiertes Clustering

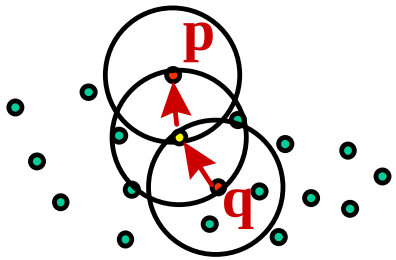
## *Grundbegriffe* [Ester, Kriegel, Sander & Xu 1996]

- Ein Objekt o $\in O$ heißt *Kernobjekt*, wenn gilt:

  $|N_\varepsilon(o)| \geq MinPts$, wobei $N_\varepsilon(o) = \{o' \in O \mid dist(o, o') \leq \varepsilon\}$.



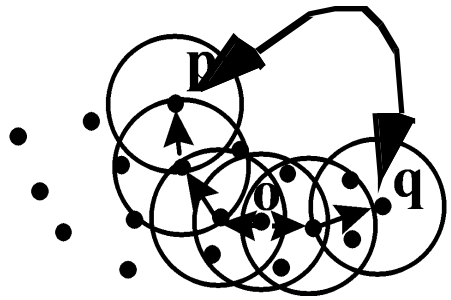

- Ein Objekt $p \in O$ ist *direkt dichte-erreichbar* von $q \in O$ bzgl. $\varepsilon$ und *MinPts*, wenn gilt: $p \in N_\varepsilon(q)$ und $q$ ist ein Kernobjekt in $O$.
- Ein Objekt $p$ ist *dichte-erreichbar* von $q$, wenn es eine Kette von direkt erreichbaren Objekten zwischen $q$ und $p$ gibt.



# 3.2 Dichtebasiertes Clustering

## *Grundbegriffe*

- 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*, 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*.

# 3.2 Dichtebasiertes Clustering

## *Grundbegriffe*

- 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*$\}$.

# 3.2 Dichtebasiertes Clustering

## *Algorithmus DBSCAN*

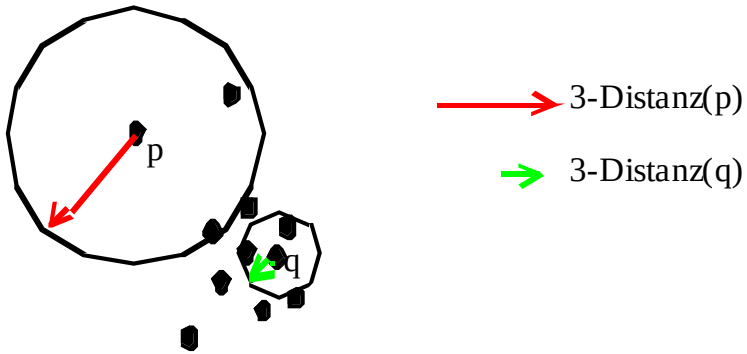
```
DBSCAN(Objektmenge D, Real ε, Integer MinPts)
  // Zu Beginn sind alle Objekte unklassifiziert,
  // o.ClId = UNKLASSIFIZIERT für alle o ∈ Objektmenge

  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);
```

# 3.2 Dichtebasiertes Clustering

## *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.

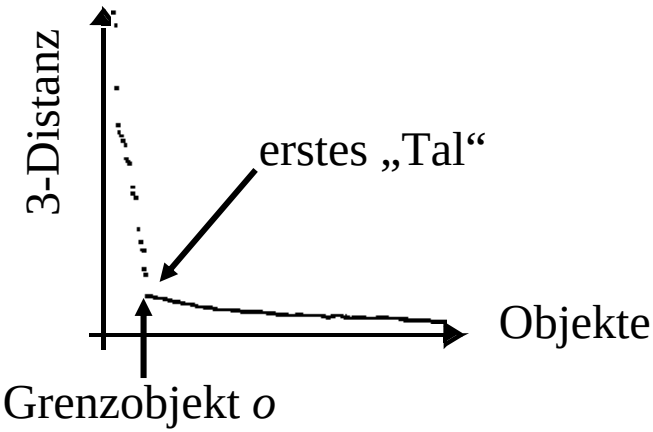



- Funktion *k-Distanz*: Distanz eines Objekts zu seinem *k*-nächsten Nachbarn
- *k-Distanz-Diagramm*: die *k*-Distanzen aller Objekte absteigend sortiert

# 3.2 Dichtebasiertes Clustering

## *Parameterbestimmung*

Beispiel eines *k*-Distanz-Diagramms



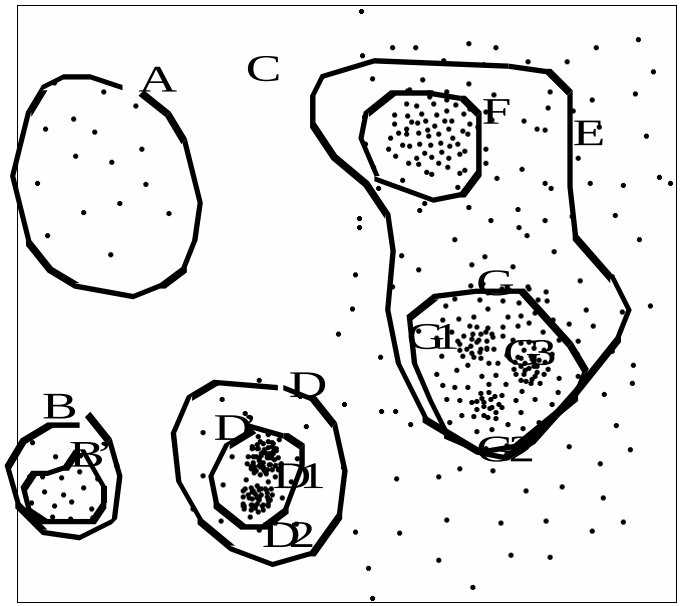

Heuristische Methode

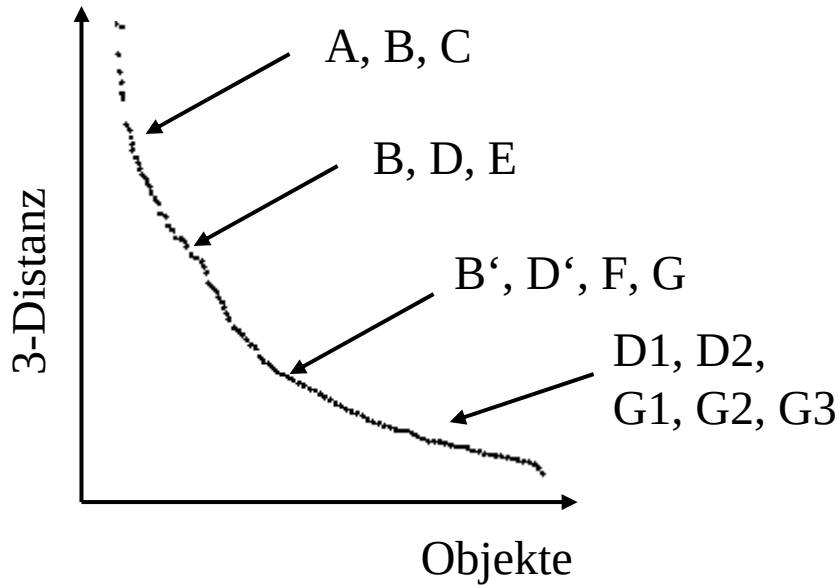
- Benutzer gibt einen Wert für *k* vor (Default ist $k = 2*d - 1$), *MinPts* := *k*+1.
- System berechnet das *k*-Distanz-Diagramm und zeigt das Diagramm an.
- Der Benutzer wählt ein Objekt *o* im *k*-Distanz-Diagramm als Grenzobjekt aus, $\varepsilon$ := *k*-Distanz(*o*).

# 3.2 Dichtebasiertes Clustering

## *Probleme der Parameterbestimmung*

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



# 3.3 Hierarchische Verfahren

## *Grundlagen*

### Ziel

Konstruktion einer Hierarchie von Clustern (*Dendrogramm*), so daß immer die Cluster mit minimaler Distanz verschmolzen werden

### Dendrogramm

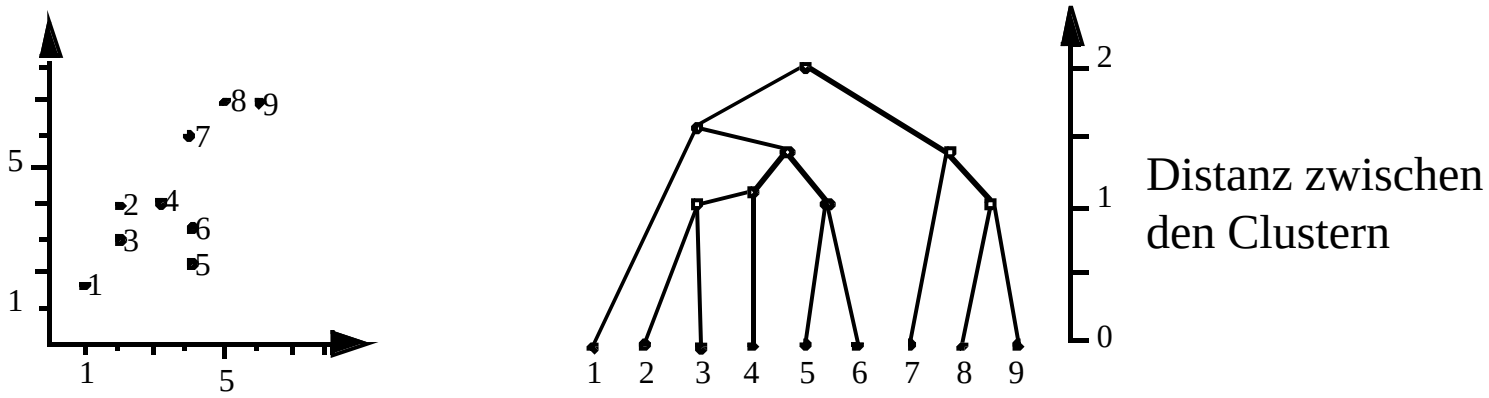
ein Baum, dessen Knoten jeweils ein Cluster repräsentieren, mit folgenden Eigenschaften:

- die Wurzel repräsentiert die ganze DB
- die Blätter repräsentieren einzelne Objekte
- ein innerer Knoten repräsentiert die Vereinigung aller Objekte, die im darunterliegenden Teilbaum repräsentiert werden

# 3.3 Hierarchische Verfahren

## *Grundlagen*

Beispiel eines Dendrogramms




Typen von hierarchischen Verfahren

- Bottom-Up Konstruktion des Dendrogramms (*agglomerative*)
- Top-Down Konstruktion des Dendrogramms (*divisive*)

# 3.3 Single-Link und Varianten

## *Algorithmus Single-Link* [Jain & Dubes 1988]

Agglomeratives hierarchisches Clustering

1. Bilde initiale Cluster, die jeweils aus einem Objekt bestehen,
   und bestimme die Distanzen zwischen allen Paaren dieser Cluster.

2. Bilde einen neuen Cluster aus den zwei Clustern,
   welche die geringste Distanz zueinander haben.

3. Bestimme die Distanz zwischen dem neuen Cluster und allen anderen Clustern.

4. Wenn alle Objekte sich in einem einzigen Cluster befinden:
   Fertig, andernfalls wiederhole ab Schritt 2.

# 3.3 Single-Link und Varianten

## *Distanzfunktionen für Cluster*

- Sei eine Distanzfunktion *dist*(*x*,*y*) für Paare von Objekten gegeben.
- Seien *X*, *Y* Cluster, d.h. Mengen von Objekten.

*Single-Link*
$$dist-sl(X,Y)=\min_{x\in X,y\in Y} dist(x,y)$$

*Complete-Link*
$$dist-cl(X,Y)=\max_{x\in X,y\in Y} dist(x,y)$$

*Average-Link*
$$dist-al(X,Y)=\frac{1}{|X|\cdot|Y|}\cdot\sum_{x\in X,y\in Y} dist(x,y)$$

# 3.3 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)

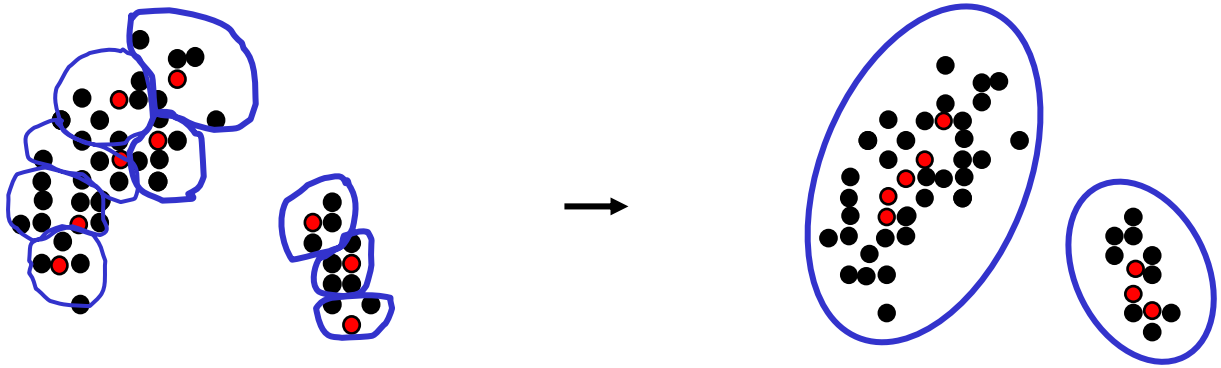
  eine „Linie“ von Objekten kann zwei Cluster verbinden
- Ineffizienz

  Laufzeitkomplexität von mindestens $O(n^2)$ für $n$ Objekte

# 3.3 Single-Link und Varianten

## *CURE* [Guha, Rastogi & Shim 1998]

- Repräsentation eines Clusters
  - partitionierende Verfahren: ein Punkt
  - hierarchische Verfahren: alle Punkte
- CURE: Repräsentation eines Clusters durch *c* Repräsentanten
- die Repräsentanten werden um den Faktor $\alpha$ zum Centroid gestreckt



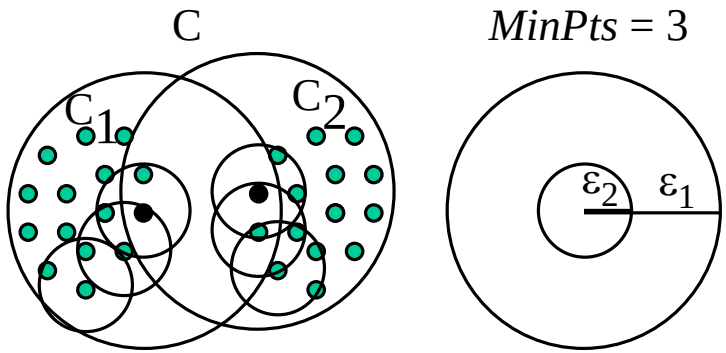
Entdecken nicht-konvexer Cluster

Vermeidung des Single-Link Effekts

# 3.3 Dichte-basiertes 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 den dichteren Teil-Cluster, dann den dünneren Rest-Cluster

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

# 3.3 Dichte-basiertes hierarchisches Clustering

## *Grundbegriffe*

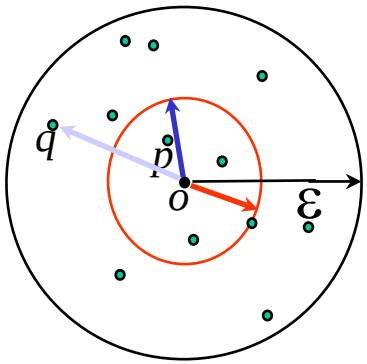
*Kerndistanz* eines Objekts *p* bzgl. ε und *MinPts*

$$Kerndis\tan z_{\varepsilon, MinPts}(o) = \begin{cases} UNDEFINIERT, \ wenn \ |N_{\varepsilon}(o)| < MinPts \\ Min-Pts-Dis\tan z(o), \ sonst \end{cases}$$

*Erreichbarkeitsdistanz* eines Objekts *p* relativ zu einem Objekt *o*

$$Erreichbarkeitsdis\tan z_{\varepsilon, MinPts}(p,o) = \begin{cases} UNDEFINIERT, \ wenn \ |N_{\varepsilon}(o)| < MinPts \\ \max\{Kerndis\tan z(o), dist(o,p)\}, \ sonst \end{cases}$$

*MinPts = 5*




*Kerndistanz*(*o*)

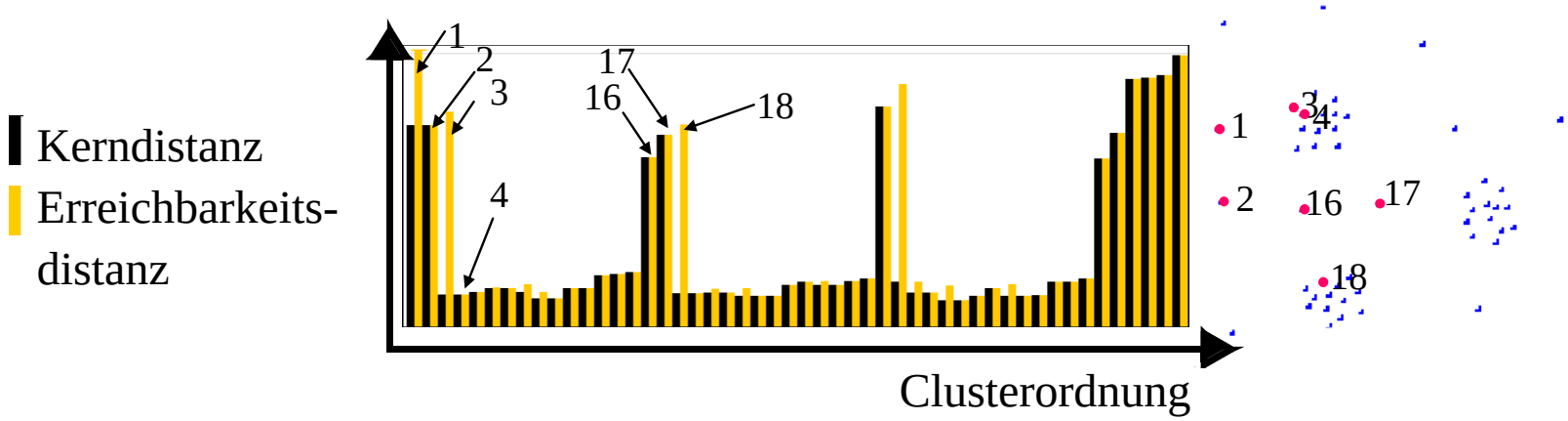
*Erreichbarkeitsdistanz*(*p*,*o*)

*Erreichbarkeitsdistanz*(*q*,*o*)

# 3.3 Dichte-basiertes hierarchisches Clustering

## *Clusterordnung*

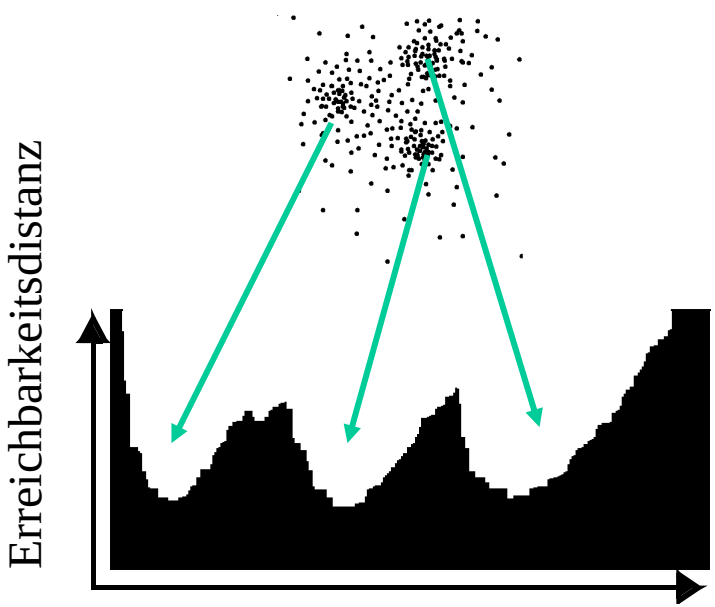
- OPTICS liefert nicht direkt ein (hierarchisches) Clustering, sondern eine „Clusterordnung“ bzgl. ε und *MinPts*
- *Clusterordnung* bzgl. ε und *MinPts*
  - beginnt mit einem beliebigen Objekt
  - als nächstes wird das Objekt besucht, das zur Menge der bisher besuchten Objekte die minimale Erreichbarkeitsdistanz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

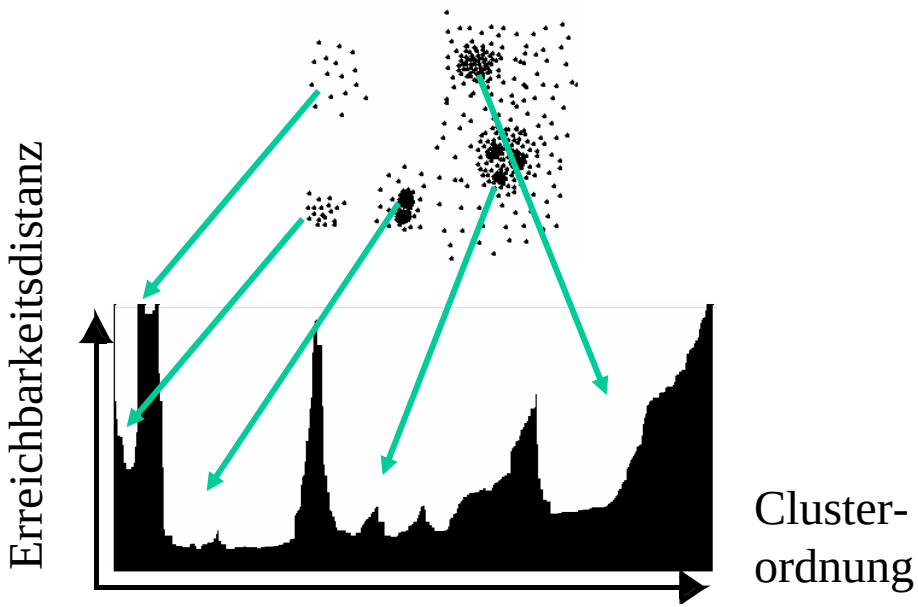


# 3.3 Dichte-basiertes hierarchisches Clustering

## *Erreichbarkeits-Diagramm*

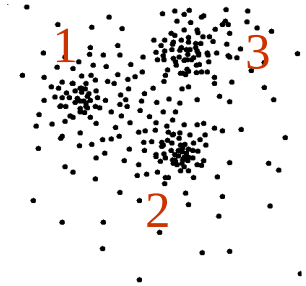
- Zeigt die Erreichbarkeitsdistanzen (bzgl. ε und *MinPts*) der Objekte als senkrechte, nebeneinander liegende Balken
- in der durch die Clusterordnung der Objekte gegebenen Reihenfolge.



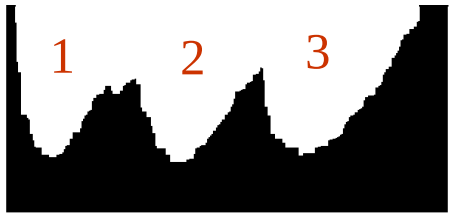



# 3.3 Dichte-basiertes hierarchisches Clustering

## *Parameter-Sensitivität*




*MinPts* = 10, ε = 10

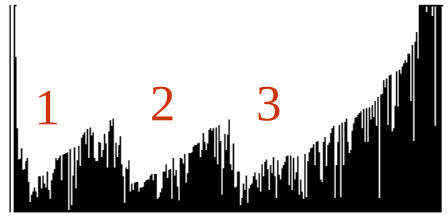



optimale Parameter

*MinPts* = 10, ε = 5


kleineres ε

*MinPts* = 2, ε = 10




kleineres *MinPts*

Clusterordnung ist robust gegenüber den Parameterwerten

gute Resultate wenn Parameterwerte „groß genug“

# 3.3 Dichte-basiertes hierarchisches Clustering

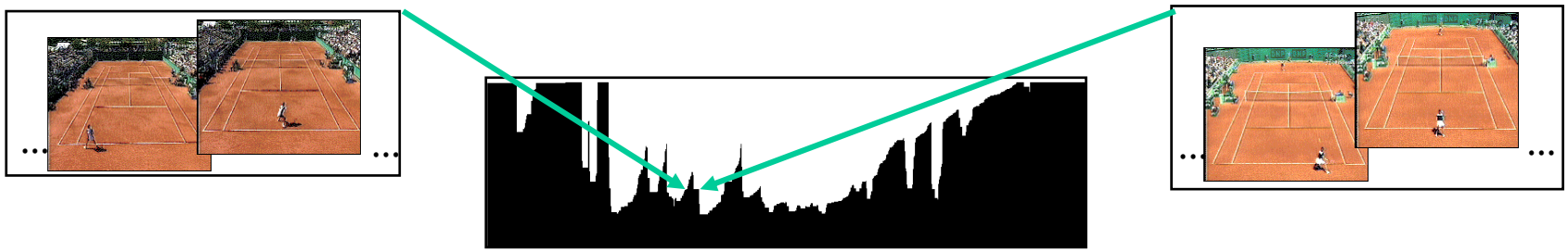
## *Heuristische Parameter-Bestimmung*

$\varepsilon$

- wähle größte *MinPts*-Distanz aus einem Sample oder
- berechne durchschnittliche *MinPts*-Distanz für gleichverteilte Daten

*MinPts*

- glätte Erreichbarkeits-Diagramm
- vermeide “single-” bzw. “*MinPts*-link” Effekt



# 3.3 Dichte-basiertes hierarchisches Clustering

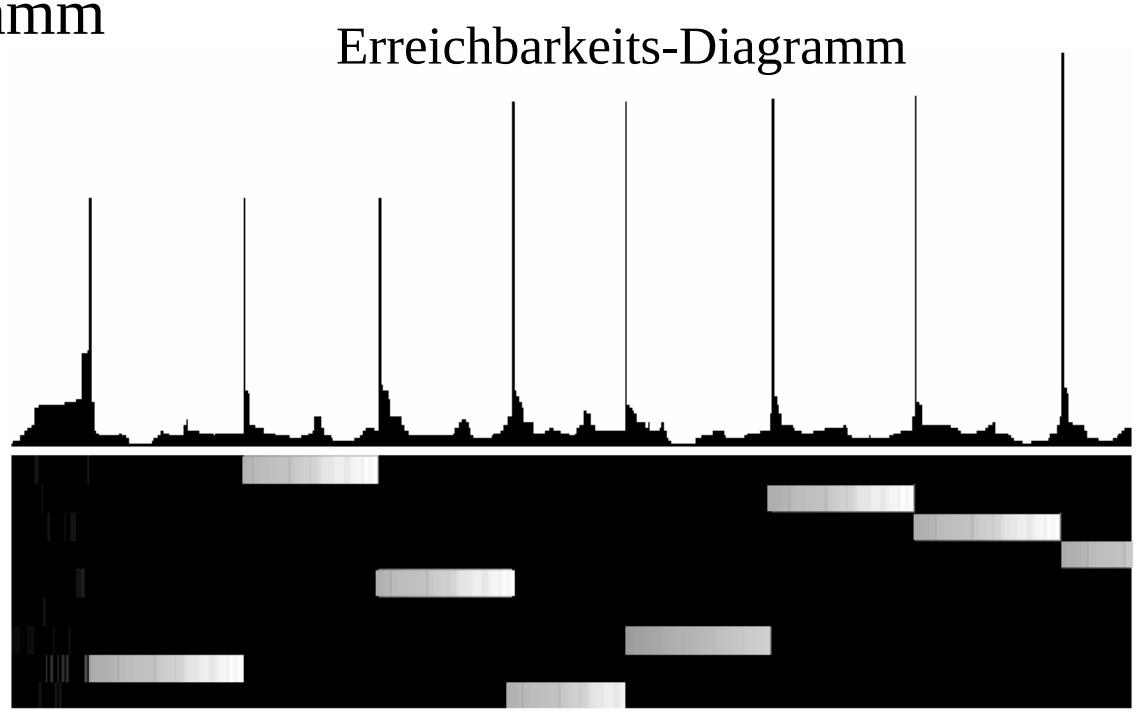
## *Manuelle Analyse der Cluster*

Mit Erreichbarkeits-Diagramm

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

Mit Attributs-Diagramm

- warum existieren die Cluster?
- worin unterscheiden sich die Cluster?



Erreichbarkeits-Diagramm

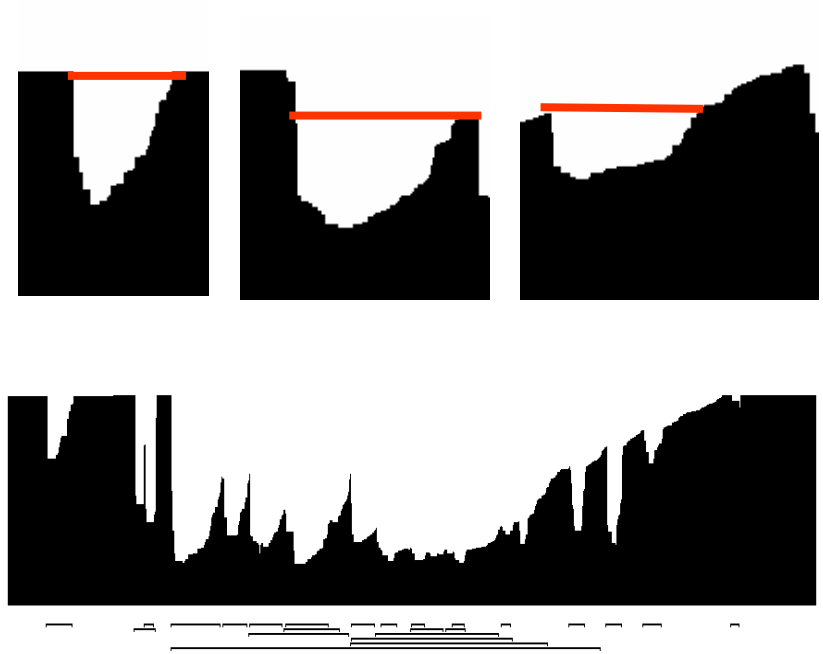
Attributs-Diagramm

# 3.3 Dichte-basiertes hierarchisches Clustering

## *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*)