Einführung in das Data Mining
Clustering / Clusteranalyse

Sascha Szott

Fachgebiet Informationssysteme
HPI Potsdam

21. Mai 2008

Teil I

Einführung

Clustering / Clusteranalyse

- Ausgangspunkt: Menge O von Objekten $\rightarrow$ einheitliche Repräsentation, z. B. als Punkte im $\mathbb{R}^n$ (*feature vectors*)
- *Ziel:* bestimme Gruppen / Klassen / Cluster $C \subseteq O$, so dass
  - zwischen Objekten innerhalb eines Clusters *möglichst große Ähnlichkeit* besteht (große intra-cluster Ähnlichkeit)
  - die Ähnlichkeit zu Objekten außerhalb des Clusters *möglichst gering* ist (geringe inter-cluster Ähnlichkeit)
- Resultat: Aufteilung der Objekte in Klassen

Clustering / Clusteranalyse

- unterschiedliche Aufteilungsstrategien
  - disjunktes Clustering $\rightarrow$ Partitionierung
  - überlappendes $\sim$: unterschiedliche Zugehörigkeitsgrade zu einzelnen Clustern (*fuzzy sets*)
  - hierarchisches $\sim \rightarrow$ Visualisierung durch Dendrogramm
  - probabilistisches $\sim$: Wahrscheinlichkeit $p(o, C)$, dass Objekt $o \in O$ zu Cluster C gehört
- Einordnung in ML: Technik des unüberwachten Lernens
  $\rightarrow$ *unlabeled data*: Klassenzugehörigkeit muss a priori nicht bekannt sein (das ist i. d. R. auch der Fall)

Allgemeines Vorgehen

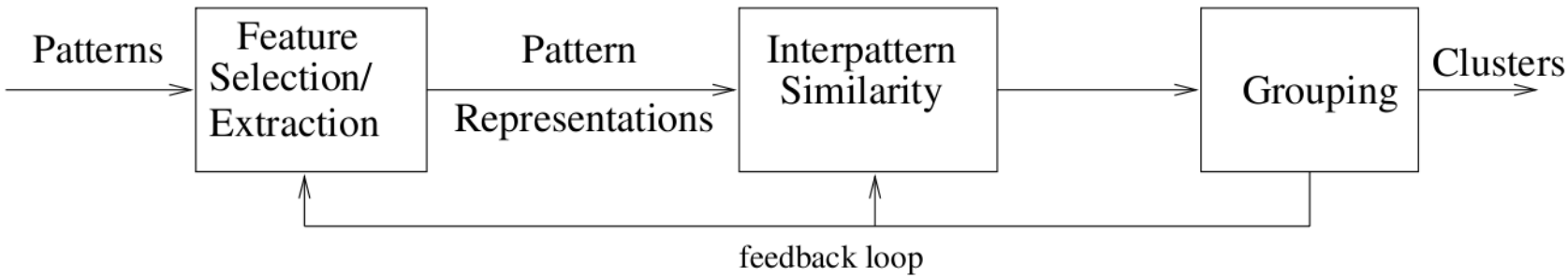
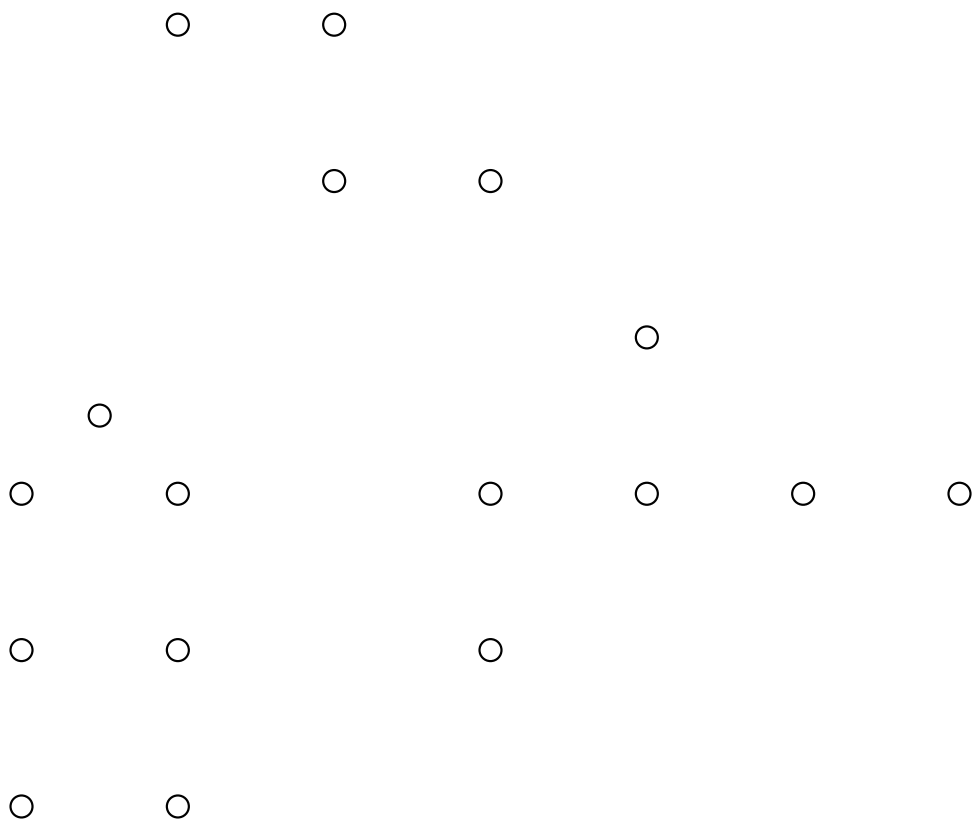


Figure 2. Stages in clustering.

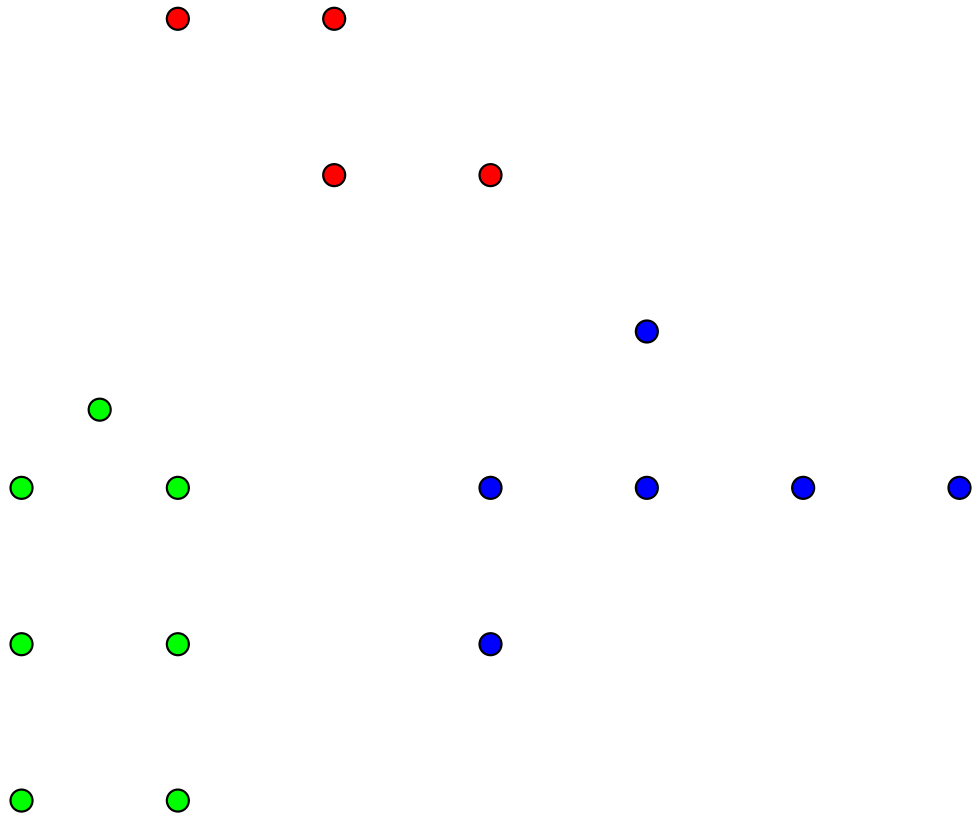
- Merkmalsauswahl → Vortrag von Frank
- Ähnlichkeitsmaße, Clusteringalgorithmen → im Folgenden

Clustering: Beispiel (hier im $\mathbb{R}^2$)



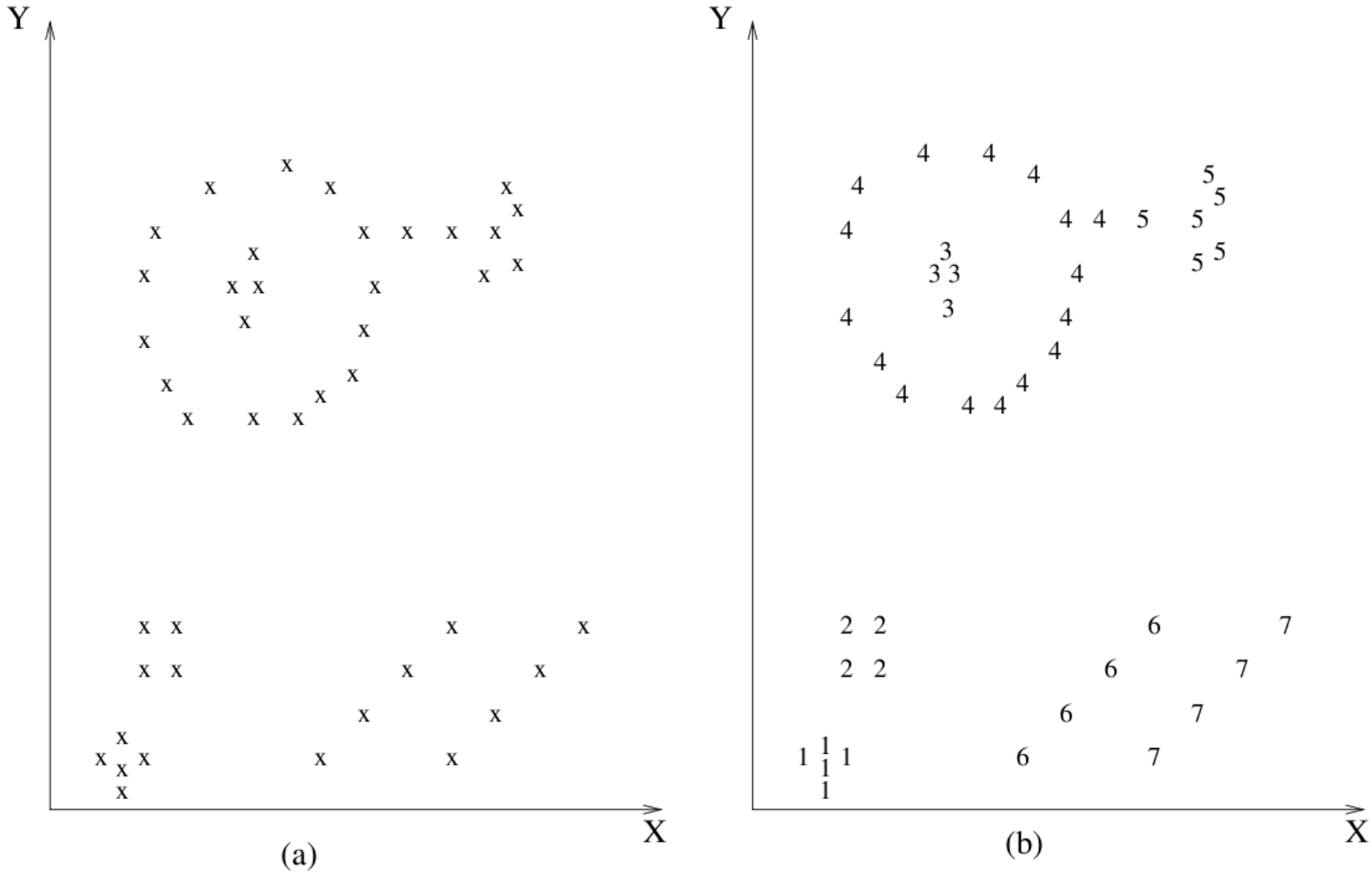
verwendetes „Ähnlichkeitsmaß“: Euklidischer Abstand
$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$

Clustering: Beispiel (hier im $\mathbb{R}^2$)



$\rightarrow$ disjunkte Aufteilung in 3 Klassen

Beispiel für nicht-konvexe Cluster



Anwendungsgebiete (in Bezug auf Ligageschichte.de)

- Typen von Fußballspielern
- Wer kann mit wem?
- Ausreißer
- Dokumentclustering (Newstexte)
- ...

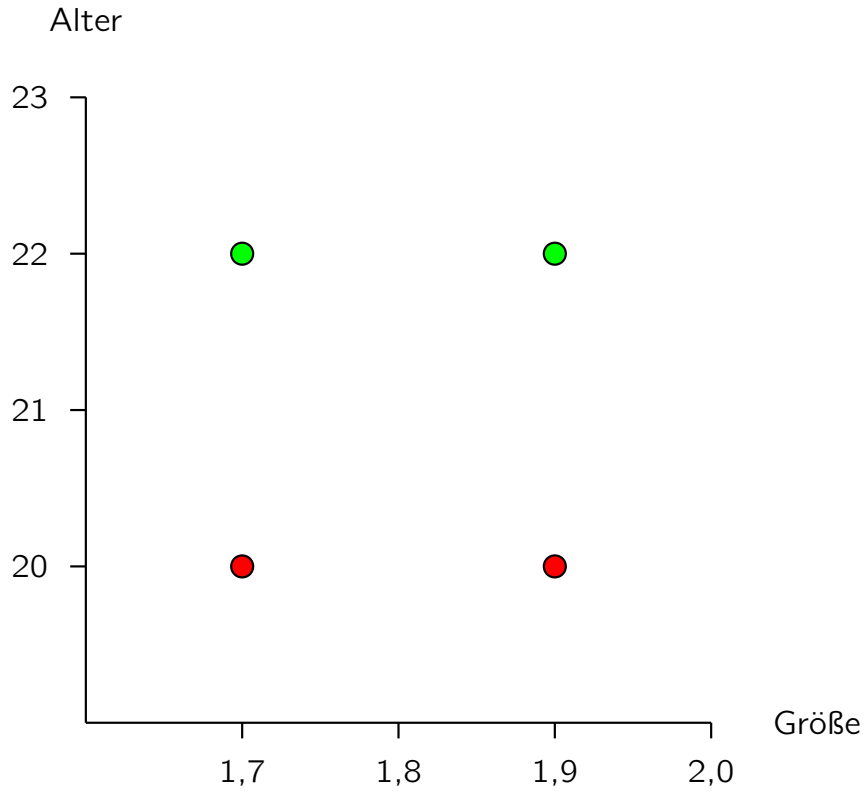
Ähnlichkeitsmaß

- zentrale Bedeutung (neben dem verwendeten Algorithmus)
- z. B. Minkowski Distanz (im $\mathbb{R}^n$):

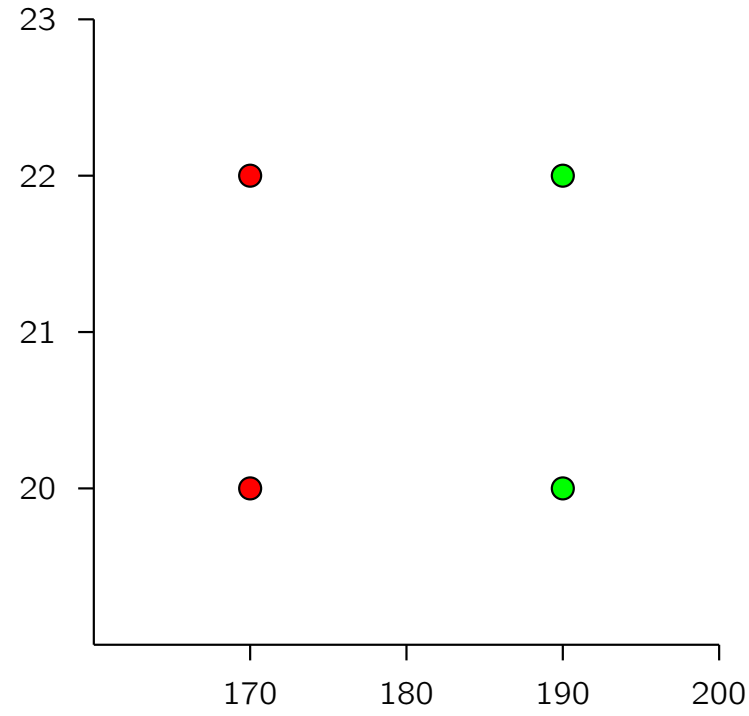
$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} = ||\mathbf{x} - \mathbf{y}||_p$$

- für $p = 1$: Manhattan Distanz
- für $p = 2$: Euklidische Distanz
- Probleme:
  - implizite Annahme, dass in jeder Dimension gleiche Skalierung
  - Behandlung von nominalen Daten (z. B. Wochentage, Spielertyp): Ordnung?, Subtraktion?
  - Berücksichtigung von Domänenwissen

Einfluss der Skalierung auf die Clusterbildung



ungünstige Skalierung
(x-Achse stauchen)



günstige Skalierung
(x-Achse dehnen)

$\rightarrow$ Abhilfe: Gewichtung, z. B. durch Normalisierung

Vorgestellte Algorithmen

disjunktes / partitionierendes Clustering

- k-means Algorithmus
- k-medoids Algorithmus
- DBSCAN-Algorithmus

hierarchisches Clustering

- top-down (aufteilendes, *divisive*) Clustering
- bottom-up (aufhäufendes, *agglomerative*) Clustering

Grenzen der Verfahren

There is no clustering technique that is universally applicable in uncovering the variety of structures present in multidimensional data sets. (Jain, Murty, Flynn. Data Clustering: A Review. 1999)

Teil II

Partitionierendes Clustering

k-means Algorithmus

Charakterisika:

- Parameter $k \in \mathbb{N}$ bestimmt die Anzahl der Cluster
- jeder Cluster C_i wird durch Zentroid $\mathbf{c}_i \in \mathbb{R}^n$ repräsentiert
  $\rightarrow$ Mittelwert bezüglich aller in C_i enthaltenen Punkte, d. h.

$$\mathbf{c}_i = \left(\frac{1}{|C_i|} \sum_{\mathbf{p} \in C_i} p_1, \ldots, \frac{1}{|C_i|} \sum_{\mathbf{p} \in C_i} p_n \right)$$

- *Ziel*: wähle Cluster $C_1, \ldots, C_k \subseteq O$ so, dass $\{C_1, \ldots, C_k\}$ eine Partition von O ist und

$$E(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{\mathbf{p} \in C_i} |\mathbf{p} - \mathbf{c}_i|^2$$

  (intra-cluster Varianz) minimiert wird

k-means Algorithmus

1. wähle k zufällige Punkte $\mathbf{c}_1, \ldots, \mathbf{c}_k \in \mathbb{R}^n$ als Zentroide
2. $\forall \mathbf{o} \in O$: ordne $\mathbf{o}$ dem nähesten Zentroid zu, d. h. $\mathbf{o}$ wird $\mathbf{c}_i$ zugeordnet, falls

$$d(\mathbf{o}, \mathbf{c}_i) = \min_{1 \leq i \leq k} d(\mathbf{o}, \mathbf{c}_i),$$

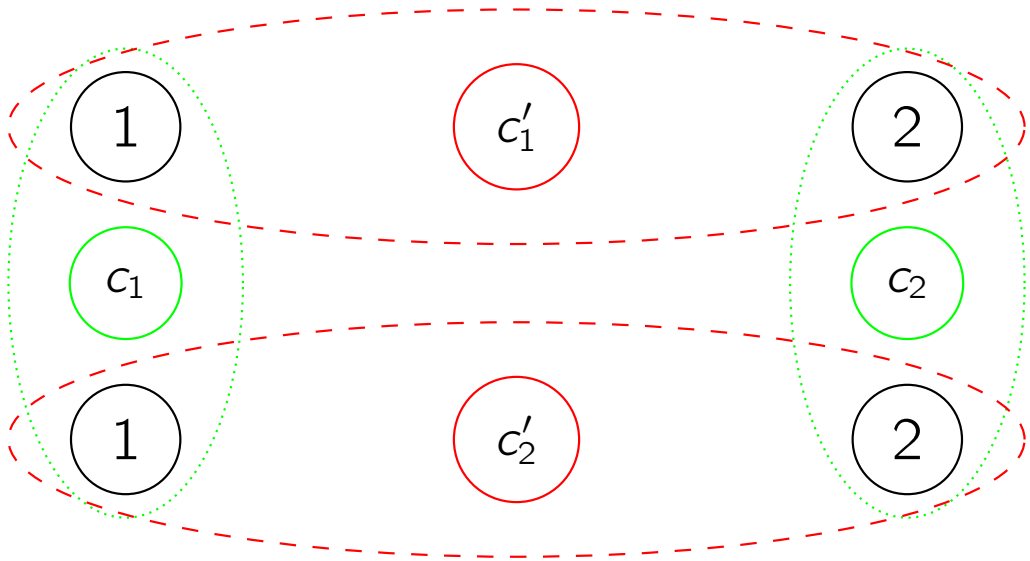
   wobei $d(,)$ eine Distanzfunktion ist
3. Sei C_i die Menge der Objekte, die $\mathbf{c}_i$ zugeordnet wurden. Berechne ausgehend von C_i den Zentroid $\mathbf{c}_i$ neu.
4. falls sich im vorherigen Schritt mindestens ein Zentroid geändert hat, gehe zu 2
   andernfalls: Stop; $C_1, \ldots, C_k$ ist eine Partitionierung von O

Demo

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

Bemerkungen zum k-means Algorithmus

Beobachtung: resultierende Cluster können stark von der Wahl der initialen Zentroide abhängen

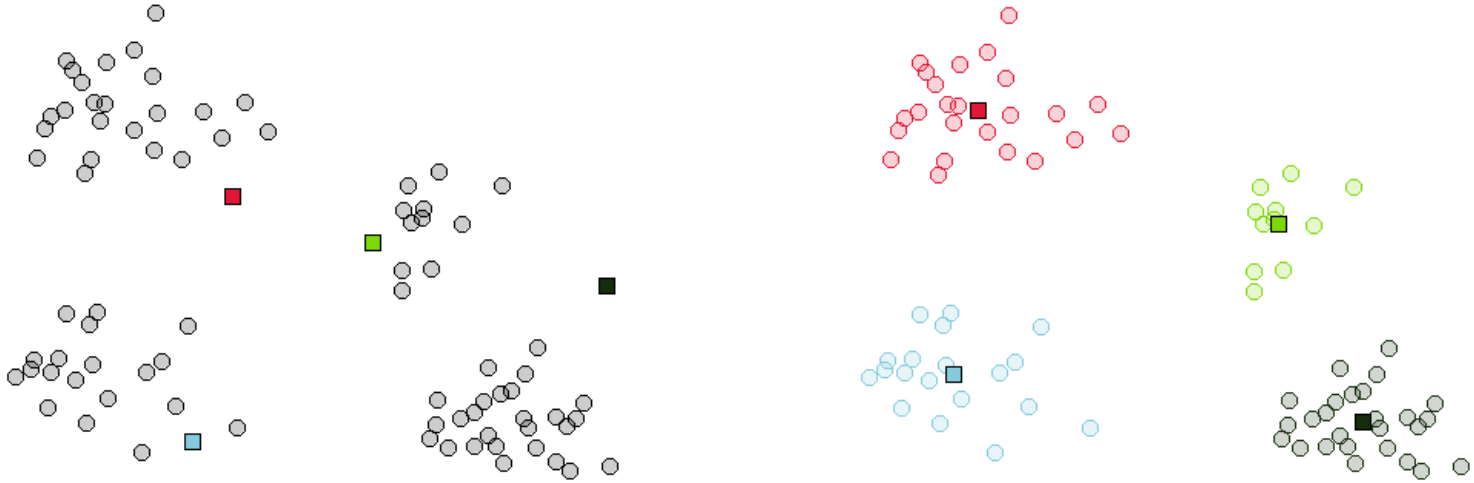


Bemerkungen zum k-means Algorithmus

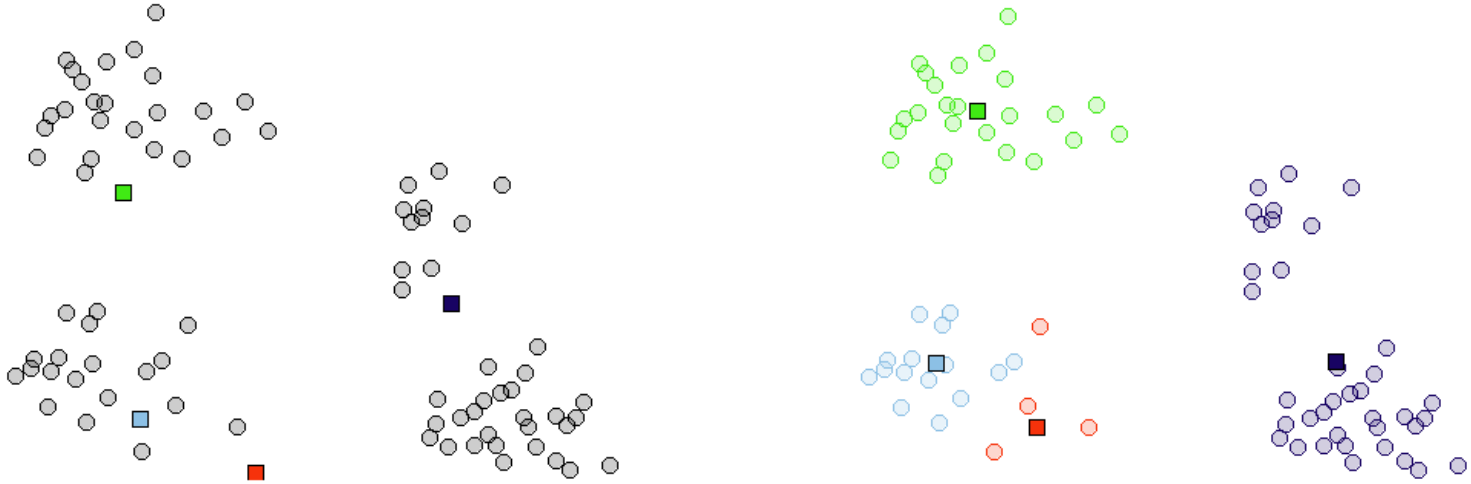
- resultierendes Clustering kann stark von Wahl der initialen Zentroide abhängen $\rightarrow$ keine Garantie dafür, dass globales Minimum bezüglich der Zielfunktion $E(C_1, \ldots, C_k)$ gefunden wird
- Abhilfe: mehrere Durchläufe mit unterschiedlichen Startkonfigurationen $\rightarrow$ wähle Resultat mit minimalen Wert für E
- genau genommen: resultierende Anzahl an Clustern kann auch kleiner k sein (Beispiel?)
- initiale Zentroide sollten möglichst weit voneinander entfernt sein

Bemerkungen zum k-means Algorithmus

gutes Resultat:



schlechtes Resultat:



Wahl des Parameters k

- Idee: bestimme Clustering für verschiedene Werte von k und wähle das Clustering mit kleinstem Wert für E
- Probleme:
  - k sollte nicht zu groß, da sonst Gefahr der Überanpassung (*overfitting*)
  - E fällt mit zunehmendem k ab

$\rightarrow$ benötigen ein von k unabhängiges Kriterium

Wahl des Parameters k

- Silhouetten-Ansatz zur Evaluierung des resultierenden Clusterings
- gegeben: Cluster $C_1, \ldots, C_k$; $C(\mathbf{o})$ sei der Cluster, der $\mathbf{o}$ enthält
- sei $a(\mathbf{o})$ die mittlere Distanz von $\mathbf{o}$ zu allen anderen Objekten in $C(\mathbf{o})$, d. h.

$$a(\mathbf{o}) = \frac{1}{|C(\mathbf{o})| - 1} \sum_{\mathbf{o}' \in C(\mathbf{o}) \setminus \{\mathbf{o}\}} d(\mathbf{o}, \mathbf{o}')$$

- sei die mittlere Distanz von $\mathbf{o}$ zu den Punkten im Cluster $C_i \neq C(\mathbf{o})$, d. h.

$$d(\mathbf{o}, C_i) = \frac{1}{|C_i|} \sum_{\mathbf{o}' \in C_i} d(\mathbf{o}, \mathbf{o}')$$

- sei $a'(\mathbf{o}) = \min\{d(\mathbf{o}, C_i) : 1 \leq i \leq k, C_i \neq C(\mathbf{o})\}$

Wahl des Parameters k

- dann ist die Silhouetten-Breite von $\mathbf{o}$ gegeben durch

$$s(\mathbf{o}) = \frac{a'(\mathbf{o}) - a(\mathbf{o})}{\max\{a(\mathbf{o}), a'(\mathbf{o})\}} \in [-1, 1]$$

- $s(\mathbf{o}) \approx 1$: $\mathbf{o}$ liegt in der Nähe der Objekte in $C(\mathbf{o})$
- $s(\mathbf{o}) \approx 0$: $\mathbf{o}$ liegt genauso weit von Objekten in $C(\mathbf{o})$ entfernt, wie von Objekten im nächstgelegenen Cluster
- $s(\mathbf{o}) \approx -1$: $\mathbf{o}$ liegt dichter an Objekten des nächstgelegenen Cluster, als an denen in $C(\mathbf{o})$

Wahl des Parameters k

- berechne durchschnittliche Clusterbreite

$$s(O) = \frac{1}{|O|} \sum_{\mathbf{o} \in O} s(\mathbf{o})$$

- nutze das Clustering, für das $s(O)$ am größten ist
- Interpretation:
  - $s(O) > 0.75$: „starke“ Struktur
  - $s(O) > 0.5$: „vernünftige“ Struktur

k-medoids Algorithmus; PAM (Partitioning Around Medoids)

Idee: für die Repräsentation von Clustern können nur Instanzen aus O gewählt werden (*Medoide*)

1. wähle k Instanzen aus O zufällig als Medoide $\mathbf{m}_1, \ldots, \mathbf{m}_k$ aus
2. $\forall \mathbf{o} \in O$: ordne $\mathbf{o}$ dem am nähesten liegenden Medoid zu
3. Sei C_i $(1 \leq i \leq k)$ die Menge der Instanzen, die $\mathbf{m}_i$ zugeordnet wurden
   $\forall C_i \; \forall \mathbf{o} \in C_i \backslash \{\mathbf{m}_i\}$: falls

$$\sum_{\mathbf{o}' \in C_i} |\mathbf{o}' - \mathbf{o}| < \sum_{\mathbf{o}' \in C_i} |\mathbf{o}' - \mathbf{m}_i|,$$

   so wähle $\mathbf{o}$ als neuen Medoid von C_i aus
4. falls in Schritt 3 mindestens ein neuer Medoid bestimmt wurde, gehe zu 2; andernfalls: Stop.

k-medoids Algorithmus; PAM (Partitioning Around Medoids)

- Ziel: Bestimme $M^* = \{\mathbf{m}_1, \ldots, \mathbf{m}_k\}$, so dass

$$M^* = \underset{M \subseteq O : |M| = k}{\operatorname{argmin}} \sum_{\mathbf{o} \in O} \min_{\mathbf{m}_i \in M} \{d(\mathbf{o}, \mathbf{m}_i)\}$$

  minimal ist
- keine Garantie, dass globales Minimum erreicht wird $\rightarrow$ wie auch k-means ein heuristisches Verfahren
- Suche nach Optimum würde prinzipiell die Aufzählung aller k-elementigen Teilmengen von O erfordern $\rightarrow$ Verbesserung: genetische Algorithmen, *simulated annealing*

Dichtebasiertes, partitionierendes Clustering

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

- Motivation: Punktdichte innerhalb eines Clusters höher als außerhalb des Clusters
- resultierende Cluster können beliebige Form haben; bei distanzbasierten Methoden ausschließlich konvexe Cluster
- Clusteranzahl k muss nicht initial vorgegeben werden

figure 1: Sample databases

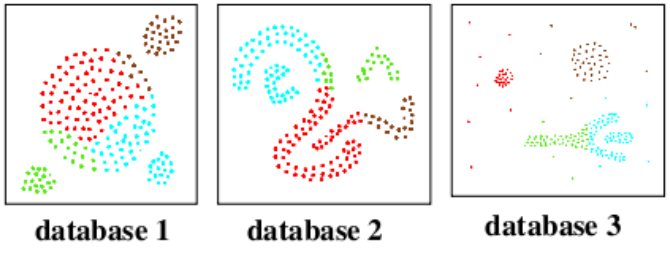


figure 5: Clusterings discovered by CLARANS

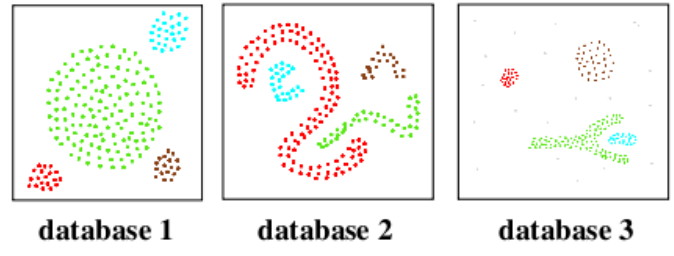


figure 6: Clusterings discovered by DBSCAN

(CLARANS: k-medoid Algorithmus)

DBSCAN – Definitionen

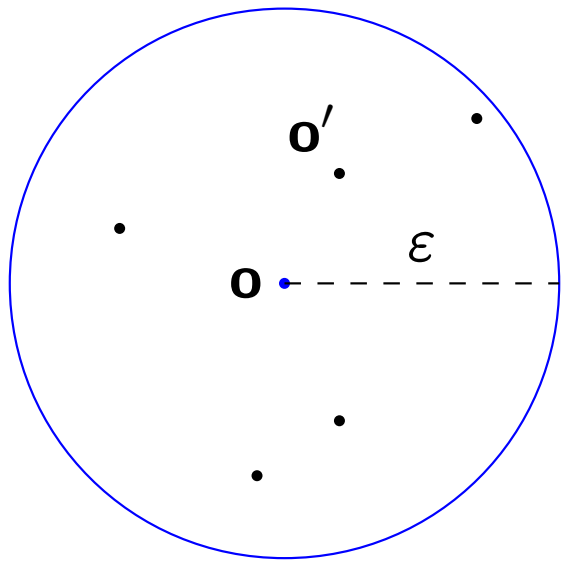
- ε-*Nachbarschaft* eines Objektes $\mathbf{o} \in O$:

$$N_\varepsilon(\mathbf{o}) := \{\mathbf{o}' \in O : d(\mathbf{o}, \mathbf{o}') \leq \varepsilon\}$$

- Aufteilung der Objekte in O
  - $\mathbf{o} \in O$ heißt *Kernobjekt* $:\Longleftrightarrow |N_\varepsilon(\mathbf{o})| \geq m$
  - $\mathbf{o} \in O$ heißt *Randobjekt* $:\Longleftrightarrow \mathbf{o}$ ist kein Kernobjekt
- Parameter $\varepsilon \in \mathbb{R}^+$ und $m \in \mathbb{N}$ müssen initial vorgegeben werden (Heuristik zur Bestimmung der Parameter basierend auf der Dichte des „dünnsten“ Clusters)
- im Folgenden sei core(O) die Menge aller Kernobjekte in O
- in den folgenden Beispielen: $m = 4$

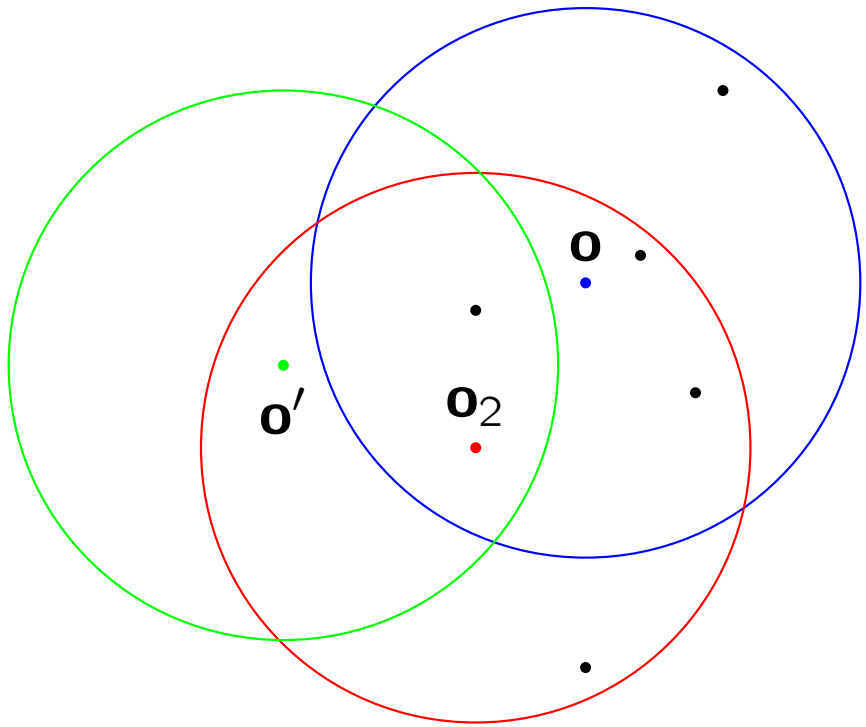
DBSCAN – Definitionen

$\mathbf{o}' \in O$ ist *direkt dichte-erreichbar* von $\mathbf{o} \in O :\Longleftrightarrow$
$\mathbf{o}' \in N_\varepsilon(\mathbf{o}) \wedge \mathbf{o} \in \text{core}(O)$



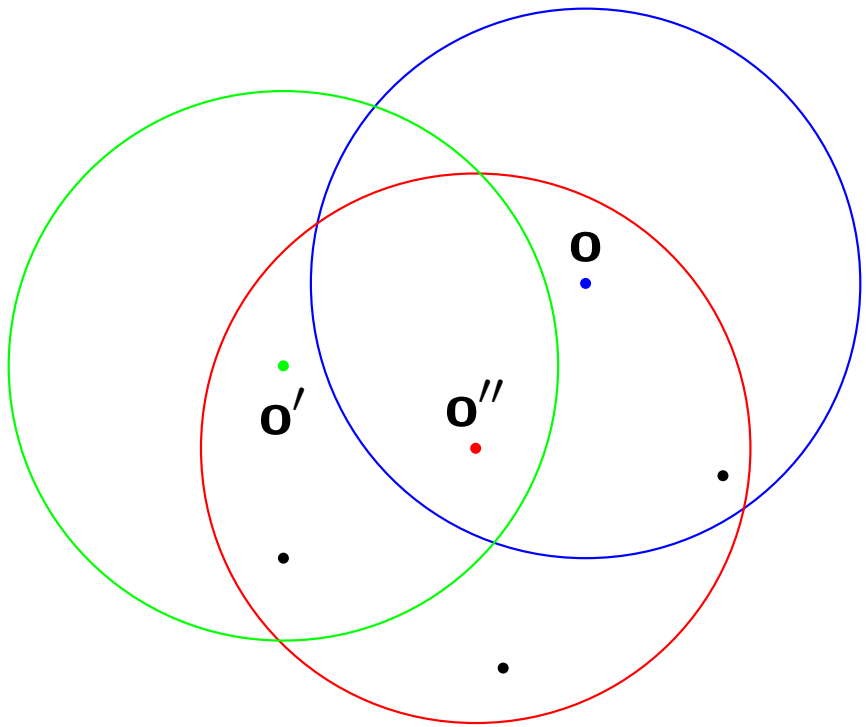
DBSCAN – Definitionen

$\mathbf{o}'$ ist *dichte-erreichbar* von $\mathbf{o}$ $:\Longleftrightarrow$
$\exists \mathbf{o}_1, \ldots, \mathbf{o}_i \in O : \mathbf{o}_1 = \mathbf{o} \wedge \mathbf{o}_i = \mathbf{o}' \wedge \forall j \in \{1, \ldots, i-1\} :$
$\mathbf{o}_{j+1}$ direkt dichte-erreichbar von $\mathbf{o}_j$



DBSCAN – Definitionen

$\mathbf{o}, \mathbf{o}' \in O$ sind *dichte-verbunden* $:\Longleftrightarrow \exists \mathbf{o}'' \in O$: $\mathbf{o}$ und $\mathbf{o}'$ sind von $\mathbf{o}''$ aus dichte-erreichbar



DBSCAN – Definitionen

Ein *Cluster* C ist eine nichtleere Teilmenge von O, die folgende Bedingungen erfüllt:

1. $\forall \mathbf{o}, \mathbf{o}' \in O$: ist $\mathbf{o} \in C$ und $\mathbf{o}'$ dichte-erreichbar von $\mathbf{o}$, dann ist $\mathbf{o}' \in C$ (Maximalität)
2. $\forall \mathbf{o}, \mathbf{o}' \in C$: $\mathbf{o}$ ist dichte-verbunden mit $\mathbf{o}'$ (Konnektivität)

Seien $C_1, \ldots, C_k$ Cluster bezüglich der Parameter (ε_i, m_i) mit $1 \leq i \leq k$. Dann ist die Menge N (*noise*) definiert als:

$$N := \{\mathbf{o} \in O : \forall i \in \{1, \ldots, k\}(\mathbf{o} \notin C_i)\}$$

N enthält also die Punkte, die keinem Cluster zugeordnet sind.

DBSCAN – Lemma 1

Lemma 1

Sei $\mathbf{o} \in \text{core}(O)$, dann ist die Menge
$\{\mathbf{o}' \in O : \mathbf{o}'$ ist dichte-erreichbar von $\mathbf{o}\}$ ein Cluster.

Bestimmung eines Clusters C in zwei Schritten

1. wähle einen beliebigen Punkt $\mathbf{o} \in \text{core}(O)$
2. ermittle die Menge P aller Objekte, die von $\mathbf{o}$ aus dichte-erreichbar sind

Dann ist $C = P \cup \{\mathbf{o}\}$.

DBSCAN – Lemma 2

Lemma 2

Sei C ein Cluster und $\mathbf{o} \in C$ ein Kernobjekt. Dann gilt folgende Gleichung

$$C = \{\mathbf{o}' \in O : \mathbf{o}' \text{ ist dichte-erreichbar von } \mathbf{o}\}.$$

Damit folgt, dass ein Cluster durch *jedes* beliebige seiner Kernobjekte eindeutig bestimmt ist.

DBSCAN-Algorithmus

Eingabe: O, ε, m
Ausgabe: Funktion $c : O \to \mathbb{N}$, die jedem Objekt eine Clusternummer zuordnet

$c\_id := 1$ // -1: unclassified, -2: noise
$\forall \mathbf{o} \in O : c(\mathbf{o}) := -1$
$\forall \mathbf{o} \in O$ do
  if $c(\mathbf{o}) = -1$ then
    if ExpandCluster$(O, \mathbf{o}, c_id, \varepsilon, m)$ then
      $c\_id := c\_id + 1$
    fi
  fi
od

Funktion: ExpandCluster

Eingabe: $O, \mathbf{o} \in O, c\_id, \varepsilon, m$
Ausgabe: Wahrheitswert true oder false

$S := neighborhood(O, \mathbf{o}, \varepsilon)$
if $|S| < m$ then // **o** ist ein Randobjekt
  $c(\mathbf{o}) := -2$
  return false
else // **o** ist ein Kernobjekt
  ...
fi

// bestimme alle Objekte, die von $\mathbf{o}$ aus dichte-erreichbar sind
$\forall \mathbf{o}' \in S : c(\mathbf{o}') := c\_id$
$S := S - \{\mathbf{o}\}$
while $S \neq \emptyset$ do
  $\mathbf{o}' := S.getElement()$
  $R := neighborhood(O, \mathbf{o}', \varepsilon)$
  if $|R| \geq m$ then
    $\forall \mathbf{o}'' \in R$ do
      if $c(\mathbf{o}'') \in \{-1, -2\}$ then
        if $c(\mathbf{o}'') = -1$ then
          $S := S \cup \{\mathbf{o}''\}$
        endif
        $c(\mathbf{o}'') := c\_id$
      fi
    od
  fi
  $S := S - \{\mathbf{o}'\}$
od
return true
fi

Heuristik für Wahl der Parameter ε und m

- $\mathbf{o}'$ heißt k-nächster Nachbar von $\mathbf{o}$, wenn er in der Auflistung der Punkte aus $O - \{\mathbf{o}\}$ in aufsteigender Entfernung zu $\mathbf{o}$ an k-ter Stelle steht
- sei $d(\mathbf{o}, k)$ die Distanz von $\mathbf{o}$ zu seinem k-nächsten Nachbarn
- defineren die Funktion $d_k : O \to \mathbb{R}$ mit $\mathbf{o} \mapsto d(\mathbf{o}, k)$
- *sorted k-dist graph* ergibt sich durch das Auftragen der Werte $d_k(\mathbf{o})$ in absteigender Reihenfolge

Heuristik für Wahl der Parameter ε und m

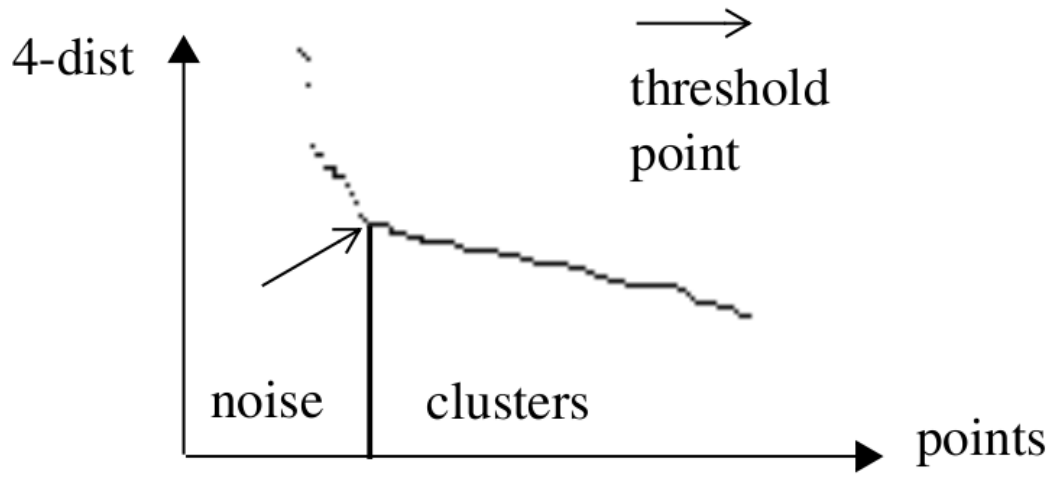


figure 4: sorted 4-dist graph for sample database 3

- wird beliebiger Punkt $\mathbf{o}$ (*threshold point*) ausgewählt, so ergibt sich
  - $\varepsilon := d_k(\mathbf{o})$ und $m := k$
  - alle Punkte $\mathbf{o}'$ mit $d_k(\mathbf{o}') \leq d_k(\mathbf{o})$ sind Kernpunkte (liegen rechts von $\mathbf{o}$ aus gesehen)
- Ziel: finde $\mathbf{o}$ mit maximalen Wert $d_k(\mathbf{o})$ im dünnsten Cluster $\rightarrow$ d. h. Punkt, der am weitesten links liegt, aber nicht zu N gehört

Heuristik für Wahl der Parameter ε und m

- heuristische Bestimmung des *threshold point*
  - verwende den ersten Punkt (von links aus gesehen), der im ersten Tal des *sorted k-dist graph* liegt
  - Punkte links davon gehören keinem Cluster an (sind in der Menge N enthalten)
  - Punkte rechts davon gehören einem Cluster an
- automatische Bestimmung des ersten Tals schwierig
  $\rightarrow$ manuelle Bestimmung durch scharfes Hinsehen (nutze $k = 4$)

Zeitkomplexität des DBSCAN-Algorithmus

- naiv: $\mathcal{O}(n^2)$ für n Eingabepunkte
- unter Verwendung von R*-Bäumen: durchschnittliche Laufzeit von $\mathcal{O}(n \log n)$
- es wurde auch eine Parallelisierung vorgeschlagen – PDBSCAN (Xu et al., A Fast Parallel Clustering Algorithm for Large Spatial Databases. 1999.)

Teil III

Hierarchisches Clustering

Hierarchisches Clustering

zwei Ansätze:

- *bottom-up (agglomerative)*: beginne mit Clustern der Größe 1 und verschmelze rekursiv zwei Cluster zu einem größeren Clustern bis sich eine bestimmte Clusteranzahl (max. 1) ergibt
- *top-down (divisive)*: beginne mit einem Cluster, das alle Objekte enthält und spalte rekursiv ein Cluster in zwei Cluster auf bis sich eine bestimmte Clusteranzahl (max. $|O|$ viele) ergibt

Visualisierung mittels *Dendrogrammen*

Hierarchisches Clustering: Dendrogramm

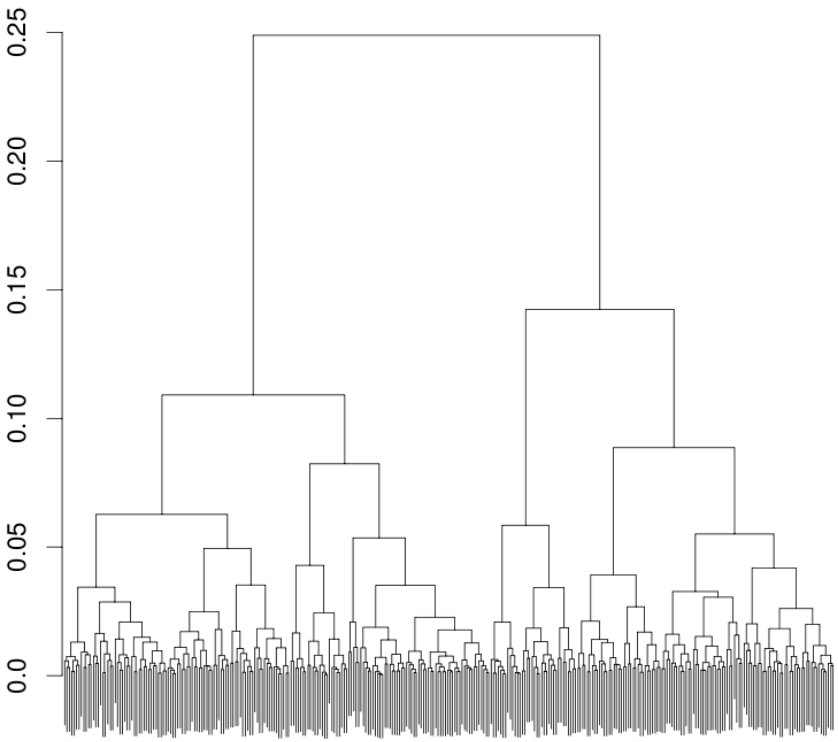


Figure 32. Hierarchical grouping of 320 views of a cobra sculpture.

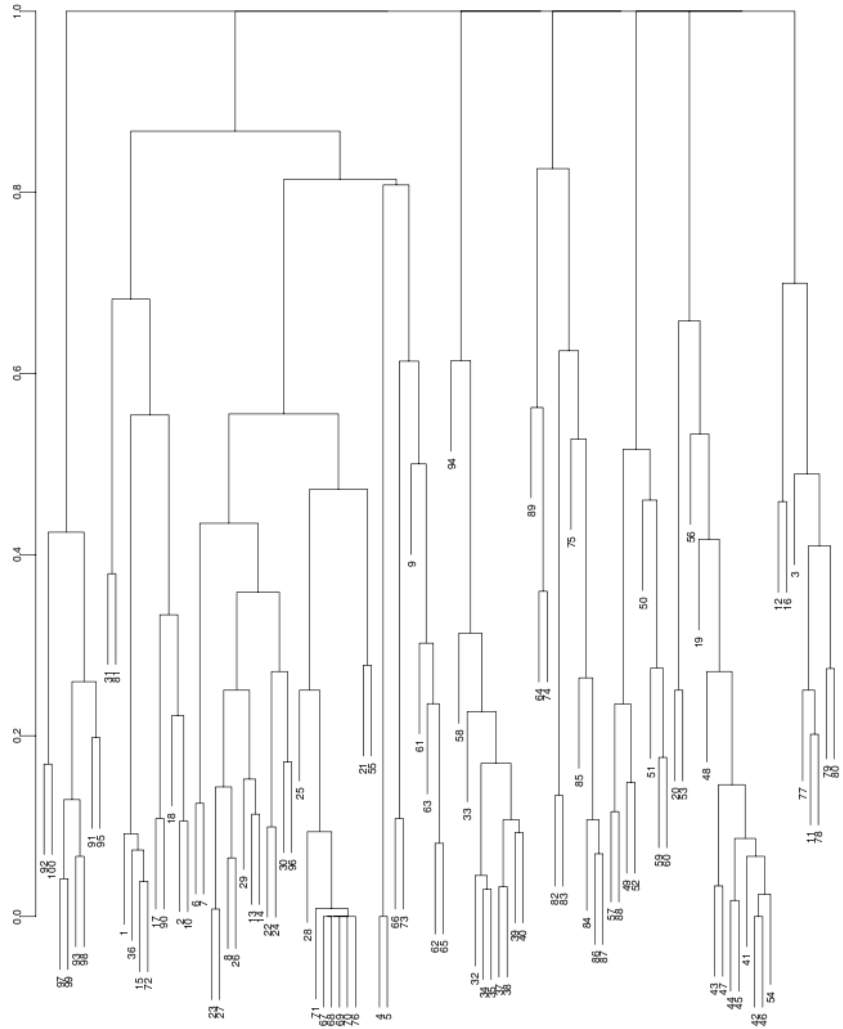


Figure 33. A dendrogram corresponding to 100 books.

Abstandsdendrogramme

- y-Achse: Abstandswert, bei dem die Cluster zusammengefügt wurden

$\rightarrow$ Maß für Ähnlichkeit der Cluster (Objekte innerhalb der Cluster)

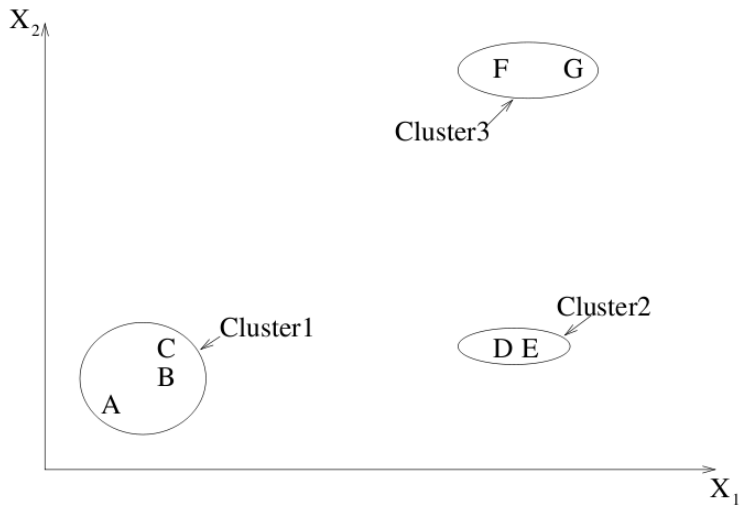


Figure 9. Points falling in three clusters.

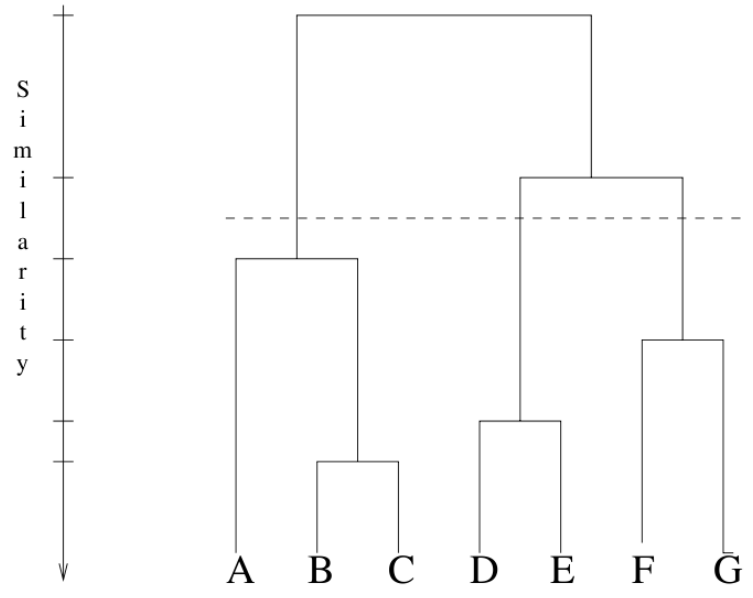


Figure 10. The dendrogram obtained using the single-link algorithm.

Bemerkung: auf y-Achse hier Ähnlichkeitswerte abgetragen

hierarchisches bottom-up Clustering

Ausgangspunkt: jedes Objekt repräsentiert ein Cluster; $d^{(0)}$ definiert Distanz zwischen je zwei Clustern

1. sei $i := 0$
2. bestimme die beiden bezüglich $d^{(i)}$ am nächsten liegenden Cluster und verschmelze diese zu einem Cluster $C \rightarrow$ die Clusteranzahl wird um 1 verringert
3. bestimme für alle anderen Cluster die jeweilige Distanz zu C; sei $d^{(i+1)}$ die sich ergebende Distanzfunktion
4. wenn die gewünschte Anzahl an Clustern erreicht ist (oder nur noch ein Cluster exitsiert): Stop; andernfalls gehe zu Schritt 2

Bemerkung: Wahl von $d^{(i+1)}$ in Schritt 3

sei C aus den Clustern C_1 und C_2 entstanden

- *single-linkage*: wähle minimalen Abstand zwischen Objekt in C und Objekt in C'

$$d^{(i+1)}(C, C') := \min\{d^{(i)}(C_1, C'), d^{(i)}(C_2, C')\}$$

- *complete-linkage*: wähle maximalen Abstand zwischen Objekt in C und Objekt in C'

$$d^{(i+1)}(C, C') := \max\{d^{(i)}(C_1, C'), d^{(i)}(C_2, C')\}$$

- *average-linkage*: wähle mittleren Abstand zwischen Objekten aus C und Objekten aus C'

$$d^{(i+1)}(C, C') := \frac{1}{2}\left(d^{(i)}(C_1, C') + d^{(i)}(C_2, C')\right)$$

Bemerkungen

Probleme:

- resultierende Cluster sind stets konvex
- unzureichende Behandlung von Ausreißern

$\rightarrow$ Abhilfe: CURE-Algorithmus (Guha et al., CURE: An Efficient Clustering Algorithm for Large Databases. Proceedings of ACM SIGMOD. 73 – 84, 1998.)

Demo

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletH.html

hierarchisches top-down Clustering: Bisecting k-means

Ausgangspunkt: ein Cluster $C = O$

1. Wähle aus den vorhandenen Clustern einen Cluster C zum Aufspalten aus
2. Spalte C in zwei Cluster unter Anwendung des k-means Algorithmus mit $k = 2$.
3. Wiederhole den zweiten Schritt r mal und wähle die Aufspaltung C_1, C_2, für die $E(C_1, C_2)$ am kleinsten ist
4. Wiederhole Schritte $1 - 3$ solange bis die gewünschte Anzahl an Clustern erreicht ist

Kriterien für die Wahl von C in Schritt 1

- größtes Cluster
- Cluster, das größtere Heterogenität aufweist
- ...

weiterführende Literatur

Jain A. K., Murty M. N., Flynn P. J.
Data Clustering: A Review.
ACM Computing Surveys. 31(3). 264 – 322. 1999.

Han J., Kamber M.
Data Mining. Concepts and Techniques.
Morgan Kaufmann. 2nd ed., 2006.

Ester M., Sander J.
Knowledge Discovery in Databases: Techniken und Anwendungen
Springer Verlag. 2000.