Data Warehousing und Data Mining

Clustering

Ulf Leser

Wissensmanagement in der Bioinformatik

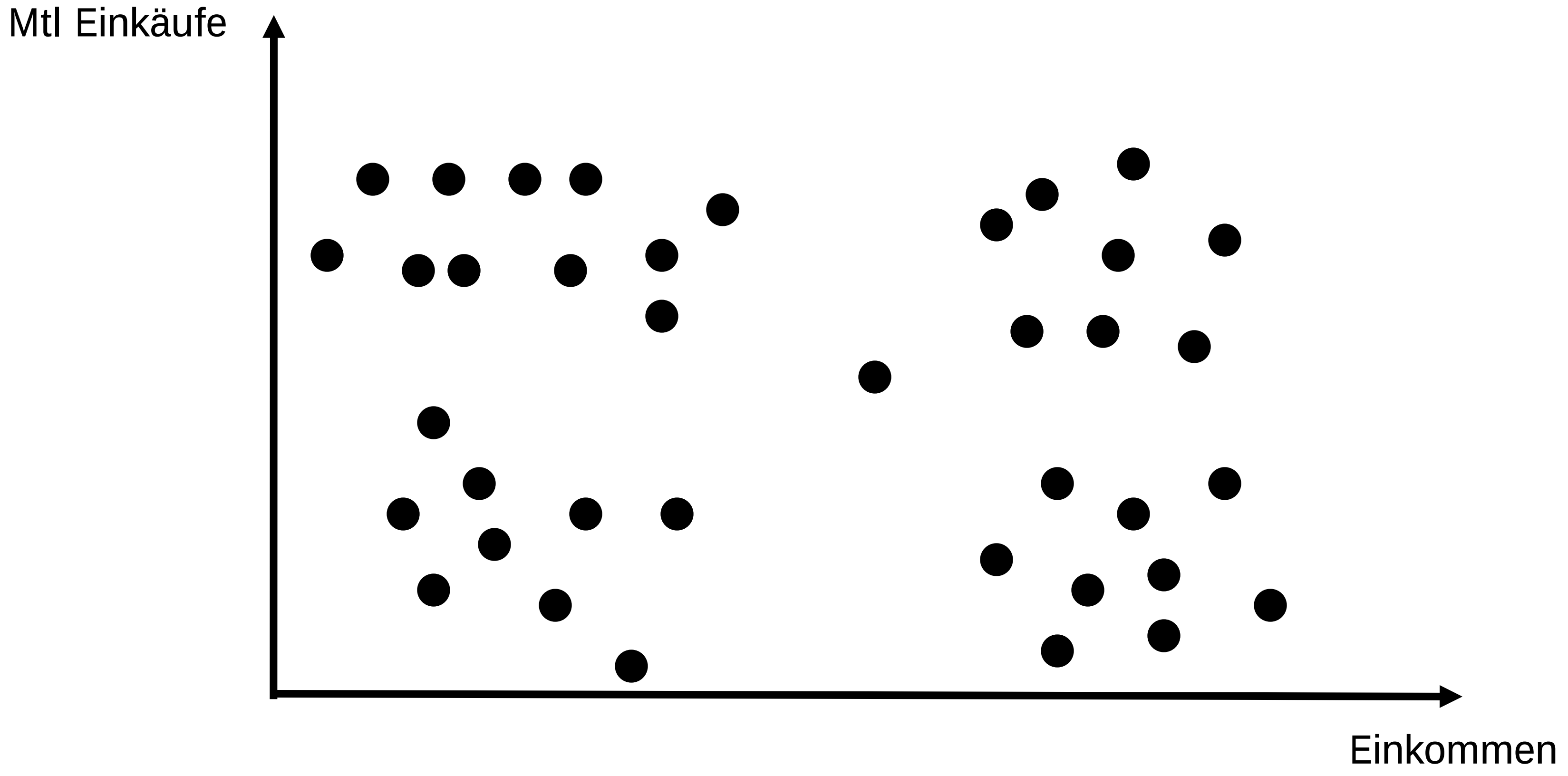
Inhalt dieser Vorlesung

- Einführung
  - Clustergüte
  - Ähnlichkeit
  - Repräsentation von Clustern
- Hierarchisches Clustering
- Partitionierendes Clustering
- Dichte-basiertes Clustering
- Weitere Verfahren

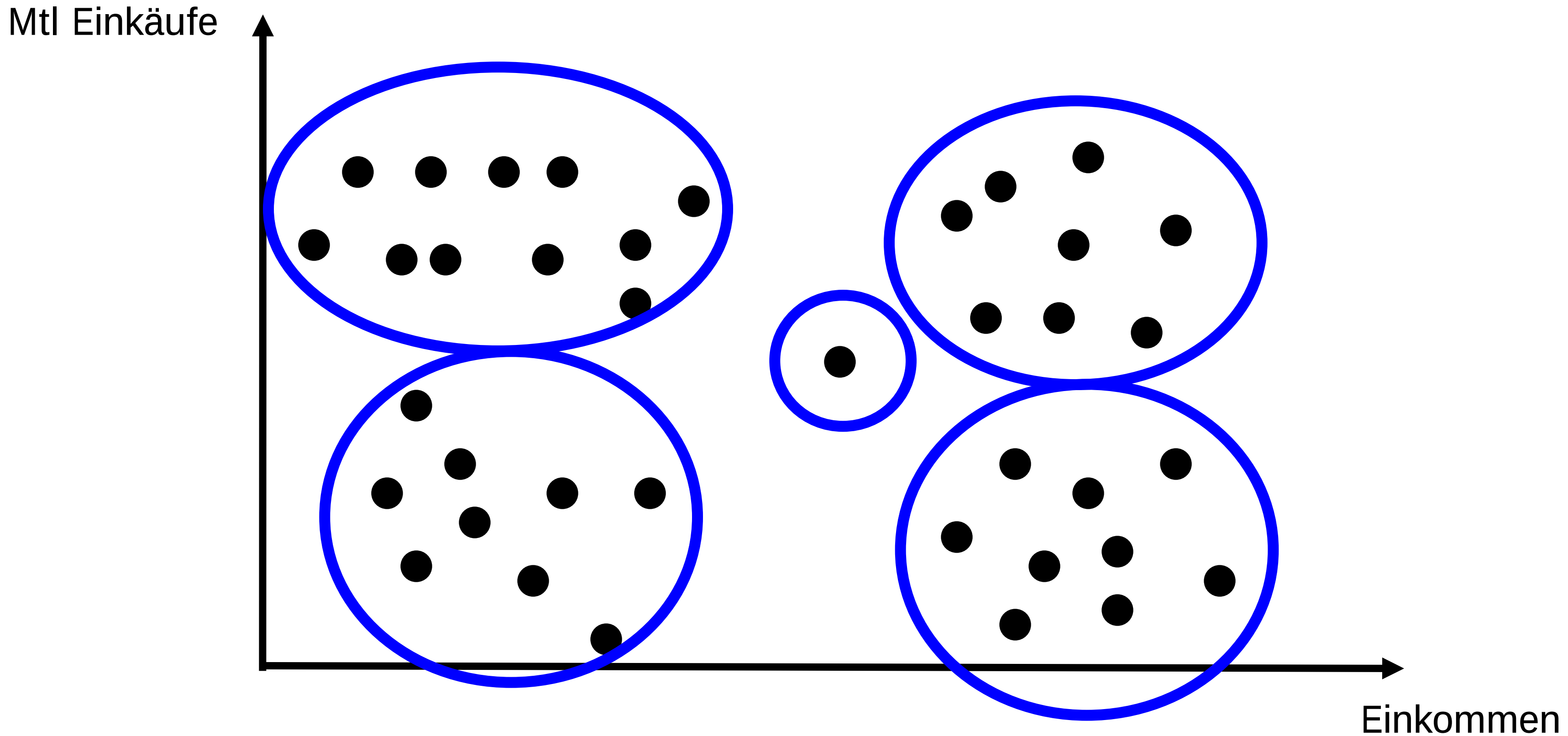
Clustering

- Finde Gruppen ähnlicher Objekte
  - Ohne zu wissen, wie viele Gruppen es geben soll
  - „Unsupervised learning“
- Anwendungen
  - Segmentiere Kunden in Gruppen (die man speziell anspricht)
  - Clustere Patienten in Verlaufsgruppen (die man speziell behandelt)
  - Finde Typen von Sternen in astronomischen Karten (die spezielle Eigenschaften haben)
  - Welche Ergebnisse einer Websuche kommen aus dem selben Thema(encluster)?
  - ...

Beispiel 1

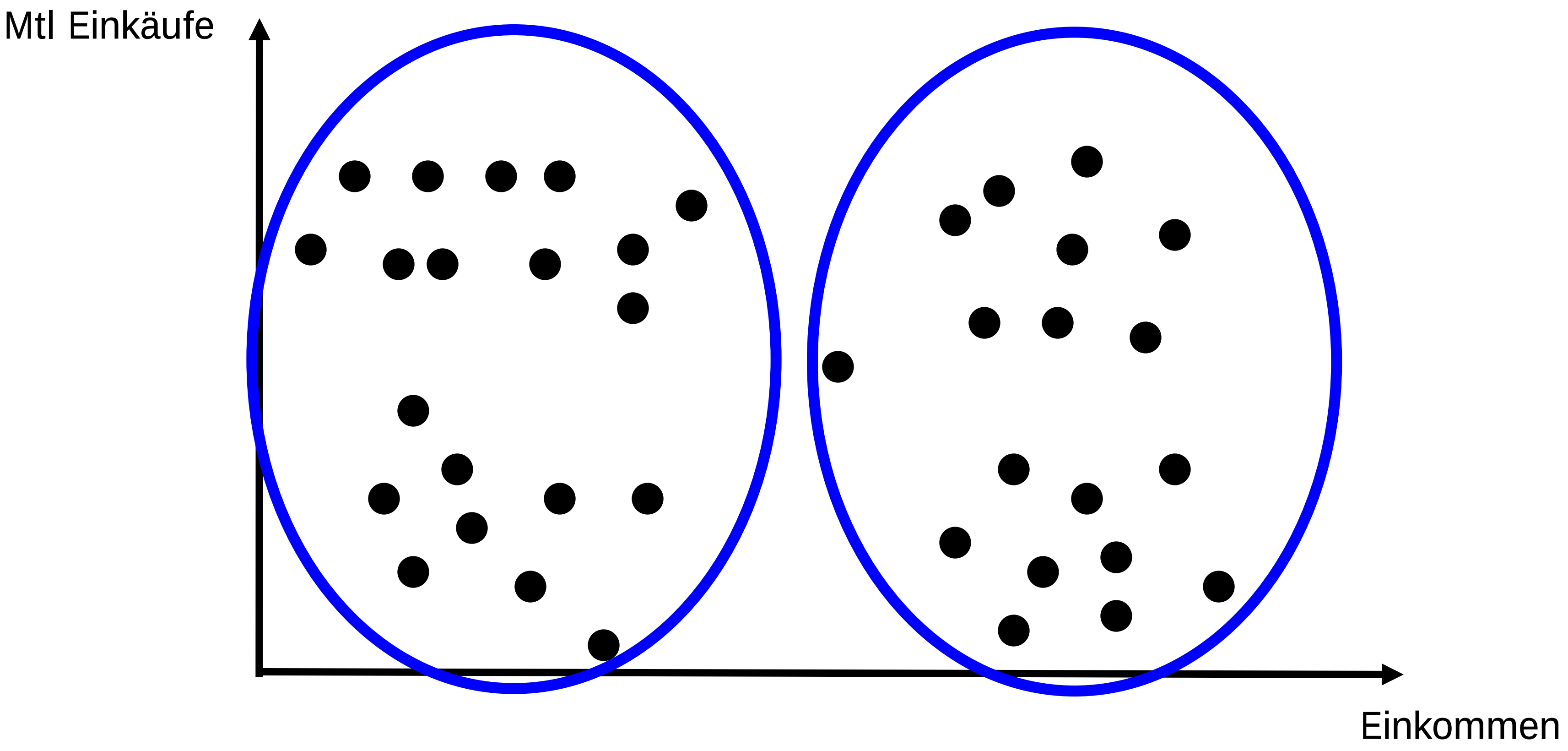


Beispiel 1



- Vier Cluster und ein Ausreißer(= eigener Cluster?)
- Überlappungsfreie, konvexe Cluster

Beispiel 2



- Zwei Cluster
- Besser?

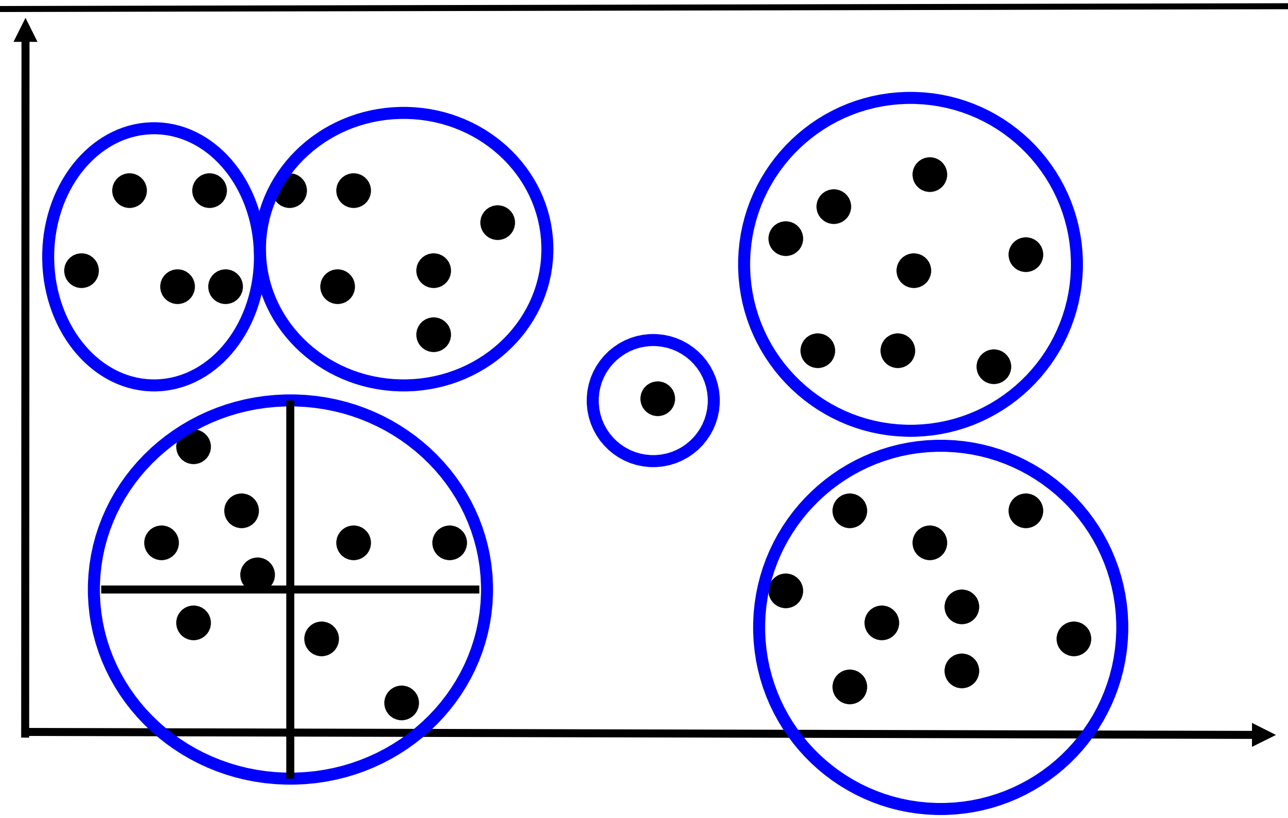
Güte eines Clusterings

- Intuitiv ist eine Gruppierung gut, wenn innerhalb jedes Clusters alle Punkte nahe beieinander liegen
- Definition
  Sei $f: O \rightarrow C$ mit $|C|=k$. Sei m_c der Mittelpunkt aller Objekte der Klasse $c \in C$, und sei $d(o,o')$ der Abstand zwischen zwei Punkten. Dann ist die k-Güte von f

$$q_k(f) = \sum_{c \in C} \sum_{f(o)=c} d(o, m_c)$$

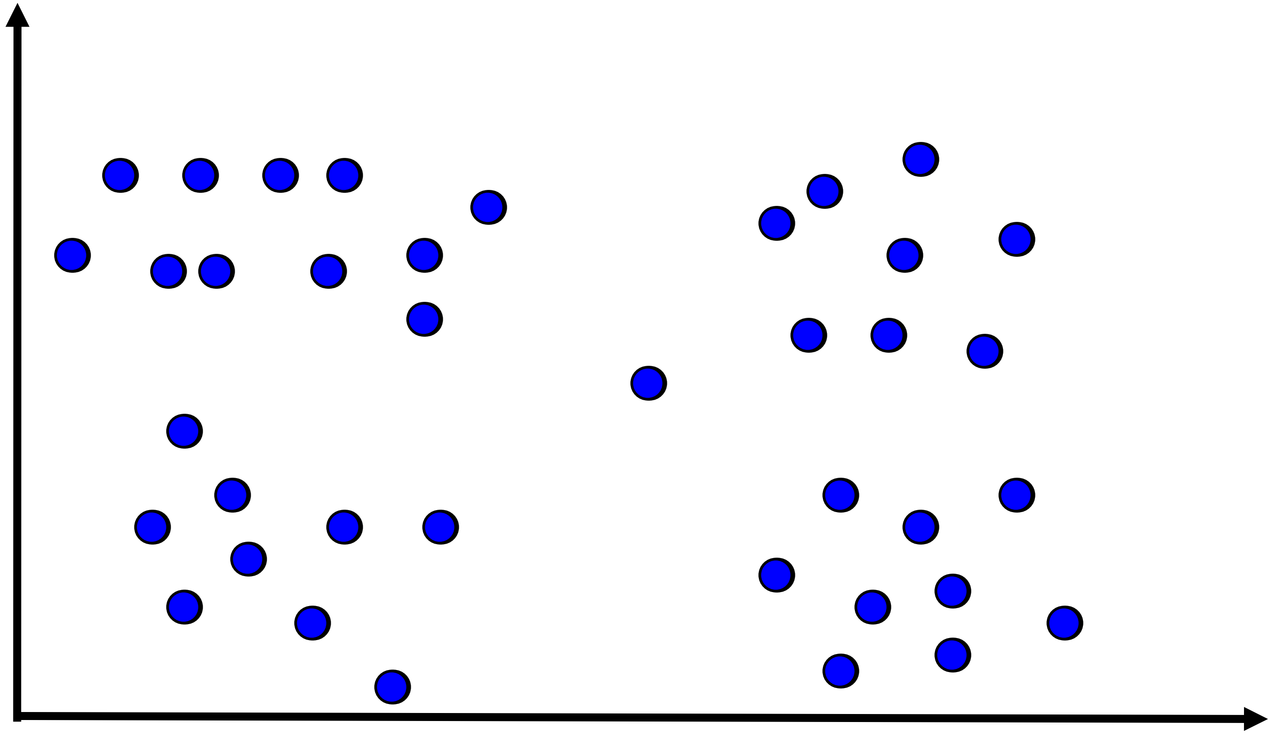
- Bemerkung
  - Zur Bestimmung von Mittelpunkten kommen wir gleich
  - Auch die Einschränkung auf k-Güte erklärt sich gleich

6-Güte



- Mittelpunkte bestimmen
- Abstand aller Punkte zu ihrem Mittelpunkt summieren
- Summe über alle Cluster

Optimales Clustering ohne Einschränkung auf k?



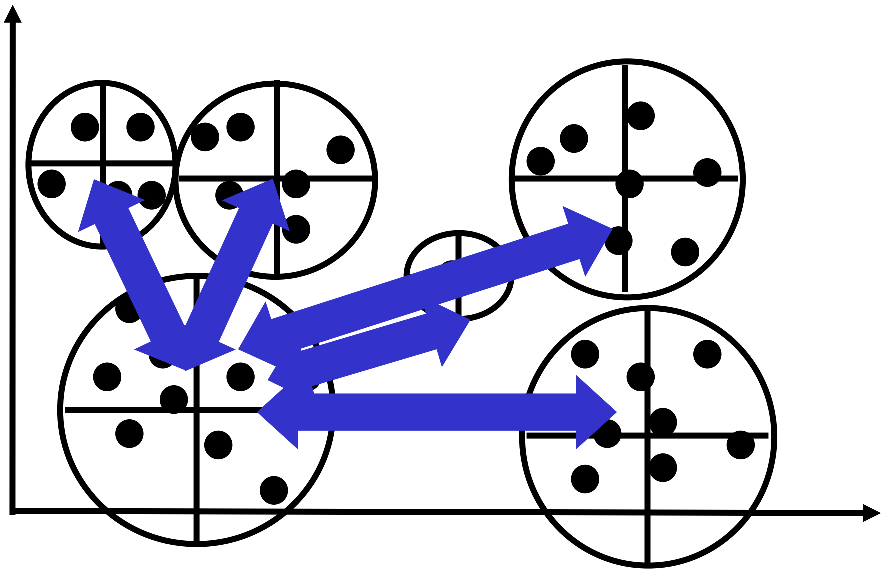
- Trivial mit k=|O|
- Score wird für größere k immer besser

Güte bei fester Anzahl von Clustern

- k-Güte ist als Maß nur dann sinnvoll, wenn die Anzahl k an Clustern vorab feststeht
- Dann ergibt sich ein Optimierungsproblem
  - Finde für eine Menge O von Objekten eine Zuordnung f in k Cluster so, dass $q_k(f)$ minimal ist
  - Aber: Problem ist NP-hart
  - Praxis: Heuristiken (z.B. k-Means)
- Score bei festem k ist sensitiv bei Ausreißern
  - Ausreisser: Punkte „weit weg“ von allen anderen
  - Bilden eigene „Cluster“ – zu weit weg von allen Mittelpunkten
  - „Normale“ Objekte müssen in weniger Cluster gepackt werden
  - Ausweg: Ausreißer vorab löschen
    - Aber wie findet man die? Clustering!

Inter/Intra-Cluster

- Bisher: Intra-Cluster Ähnlichkeit soll hoch sein
  – Geringer mittlerer Abstand
- Intuitiv soll auch die Inter-Cluster Ähnlichkeit gering sein
  – Großer Abstand jedes Punkt zu anderen Clustern
- Ein Maß, dass das berücksichtigt: Silhouette

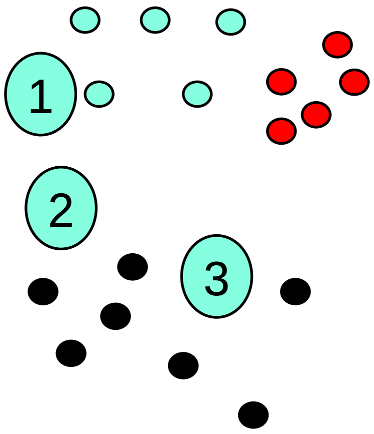


Silhouette

- Definition
  Sei f: $O \rightarrow C$ mit |C| beliebig. Sei dist(o,C_i) der mittlere Abstand von o zu allen Punkten des Clusters C_i. Dann
  - *Intra-Score: in(o) = dist(o,f(o))*
  - *Inter-Score: out(o) = min(dist(o,C_i)), $C_i \neq f(o)$*
  - *Die Silhouette s(o) eines Punktes o:*
  $$s(o) = \frac{out(o) - in(o)}{\max(in(o), out(o))}$$
  - *Die Silhouette von f ist $\Sigma s(o)$*

Eigenschaften

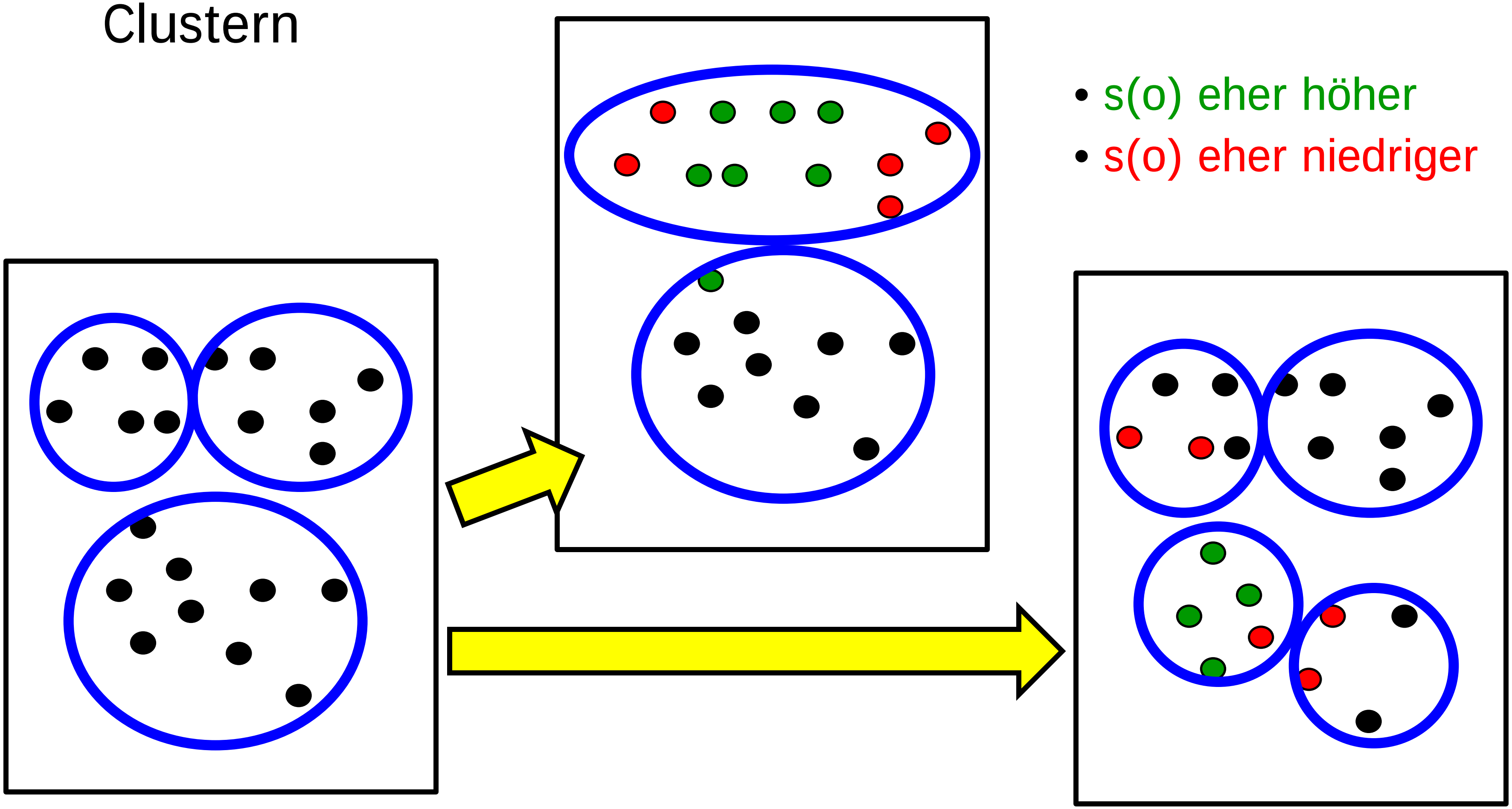
$$s(o) = \frac{out(o) - in(o)}{\max(in(o), out(o))}$$



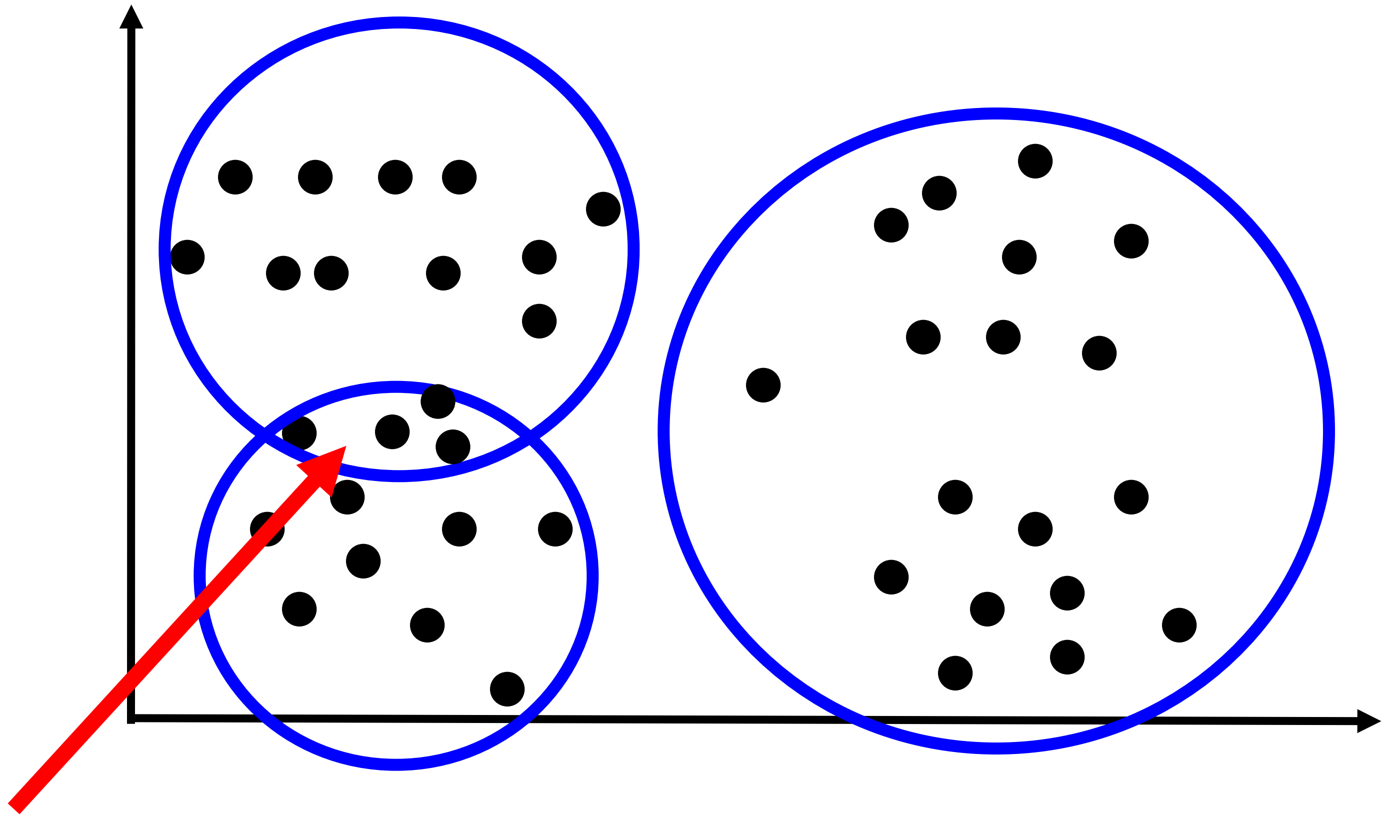
- Es gilt: $-1 \leq s(o) \leq 1$
  - s(o) ≈ 0: Punkt genau zwischen zwei Clustern (2)
  - s(o) ~ 1: Punkt sehr tief im eigenen Cluster (1)
  - s(o) ~ -1: Point viel näher an anderen Clustern (3)
- Komplexität: O(kmn)
  - Wenn Clustercentroide vorliegen und man euklidischen Abstand annimmt
  - m: Dimensionalität, n: Zahl Punkte, k: Zahl Cluster

Intuition

- Silhouette verbessert sich nicht automatisch bei mehr Clustern



Schwierige Fälle



Zu welchem Cluster sollen diese Punkte gehören?

Ähnlichkeit

- Wahl der Abstandsfunktion zwischen Objekten ist essentiell für Clusterverfahren
- Numerische Dimensionen
  - Euklidscher Abstand
    - Betont große Abstände in einzelnen Dimensionen sehr stark
    - Standard für metrische Werte
  - Cosinus-Abstand: Differenz der Winkel der Featurevektoren
    - Ausreißer in einzelnen Dimensionen zählen weniger
    - Standard z.B. beim Text-Mining
  - Normalisierung kann Dimensionen vergleichbar machen
    - Z.B. z-scores
- Kategoriale Werte: Anwendungsabhängig

Die Mitte eines Clusters

- Was ist der Mittelpunkt eines Clusters?
- Numerische Werte
  - Centroid: Mittelwert aller Punkte des Clusters
  - Medoid: Der Median aller Punkte des Clusters
    - Der „mittlerste“ Punkt von C – Punkt mit dem minimalen durchschnittlichen Abstand zu allen anderen Punkten im Cluster
    - Nachteil: Berechnung ist teuer
    - Vorteil: Weniger sensitiv bei Ausreißern
- Kategoriale Werte
  - Centroid: i.A. nicht definiert
  - Also muss man Medoid verwenden
    - Ein Abstandsmaß braucht man so oder so

Übersicht Clusteralgorithmen

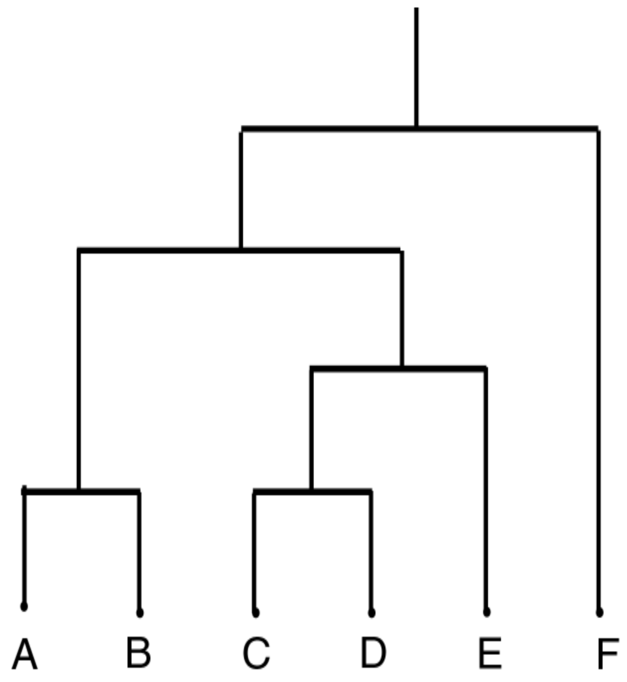
- Hierarchisch: Erzeugt hierarchisch geschachtelte Cluster
  - Benötigt kein k
  - Berechnet eigentlich keine Cluster
  - Langsam
- Partitionierend: Zerlegung der Punktmenge in k Cluster
  - Benötigt die Anzahl k der Cluster als Parameter
  - Schnell, nicht deterministisch
- Dichte-basierte: Sucht dichte Teilräume
  - Findet beliebige Bereiche mit hoher Punktdichte
  - Tendenziell sehr langsam

Inhalt dieser Vorlesung

- Einführung
- Hierarchisches Clustering
- Partitionierendes Clustering
- Dichte-basiertes Clustering
- Weitere Verfahren

Hierarchisches Clustering

- Bottom-Up Berechnung eines binären Baums (Dendrogramm)
- Algorithmus
  - Berechne Abstandsmatrix M
    - Alle $d(o_i, o_j)$, $i<j$
  - Wähle (o_i, o_j) mit $d(o_i, o_j) \overset{!}{=} \min$
  - Lösche o_i, o_j aus M
  - Füge neuen Punkt $x=<o_i,o_j>$ ein
  - Berechne Abstand von x zu allen verbleibenden Objekten/Clustern in M
    - $d(x,z) = (d(z,o_i)+d(z,o_j)) / 2$
  - Iteriere, bis M leer ist



Iteration

A
B
C
D
E
F
G

ABCDEFG
A
B .
C . .
D . . .
E . . . .
F . . . . .
G . . . . . .

$\boxed{(B,D) \rightarrow a}$

A
B
C
D
E
F
G

ACEFGa
A
C .
E . .
F . . .
G . . . .
a . . . . .

$\boxed{(E,F) \rightarrow b}$

A
B
C
D
E
F
G

ACGab
A
C .
G . .
a . . .
b . . . .

$\boxed{(A,b) \rightarrow c}$

A
B
C
D
E
F
G

CGac
C
G .
a . .
c . . .

$\boxed{(C,G) \rightarrow d}$

A
B
C
D
E
F
G

acd
a
c .
d . .

$\boxed{(d,c) \rightarrow e}$

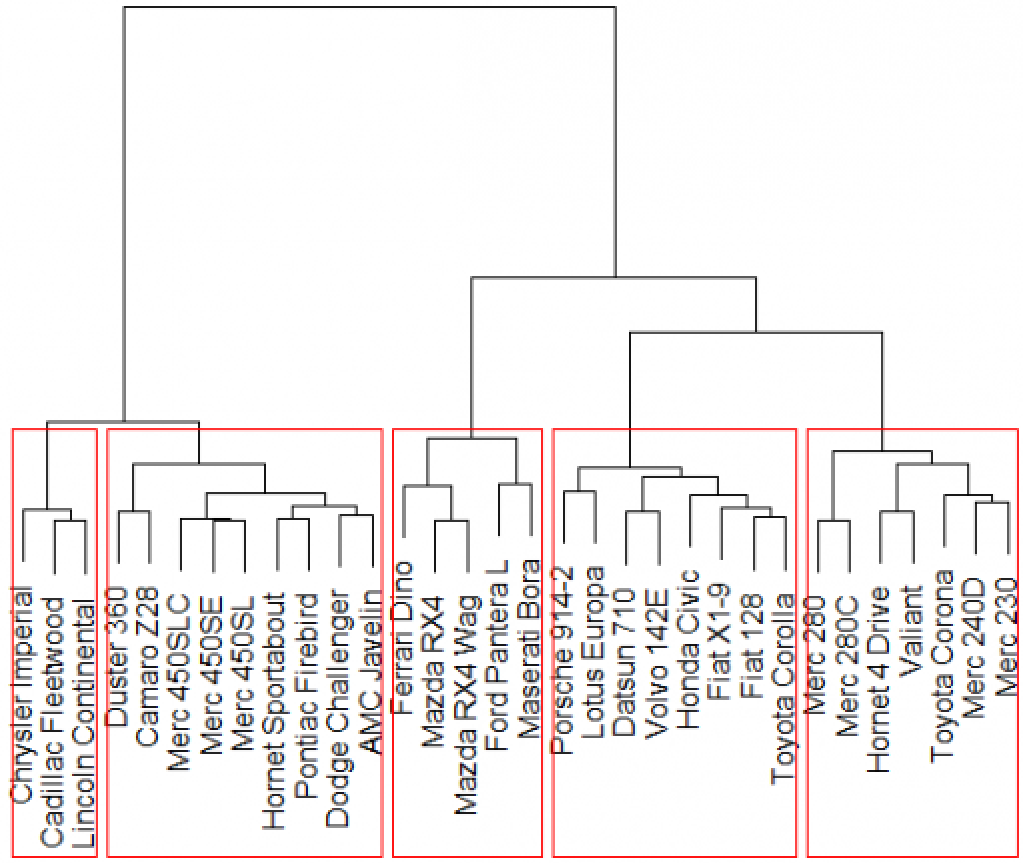
A
B
C
D
E
F
G

ae
a
e .

$\boxed{(a,e) \rightarrow f}$

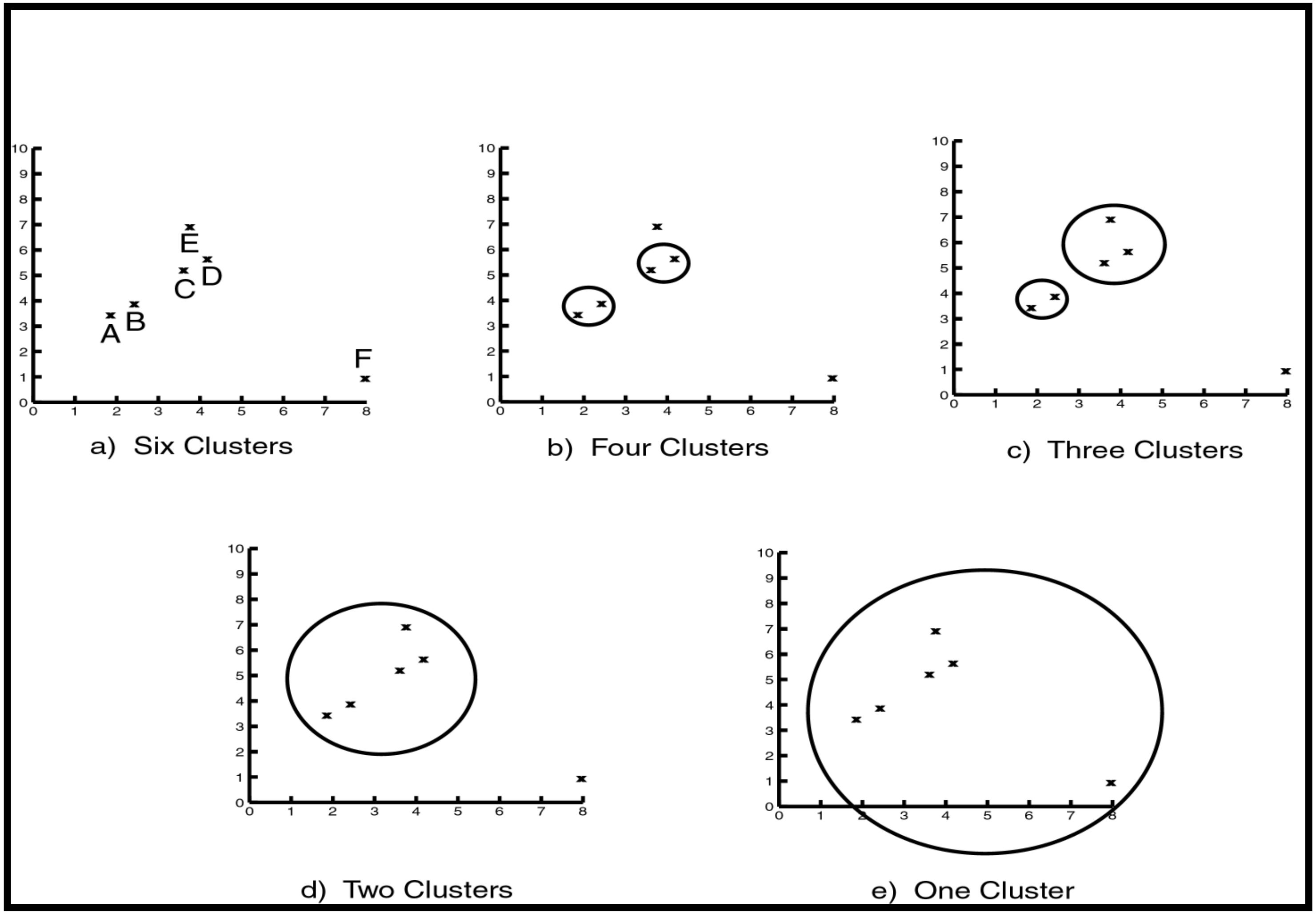
A
B
C
D
E
F
G

Bespiel: Kunden nach Automarke



Quelle: http://www.select-statistics.co.uk/article/blog-post/customer-segmentation

Hierarchische Cluster

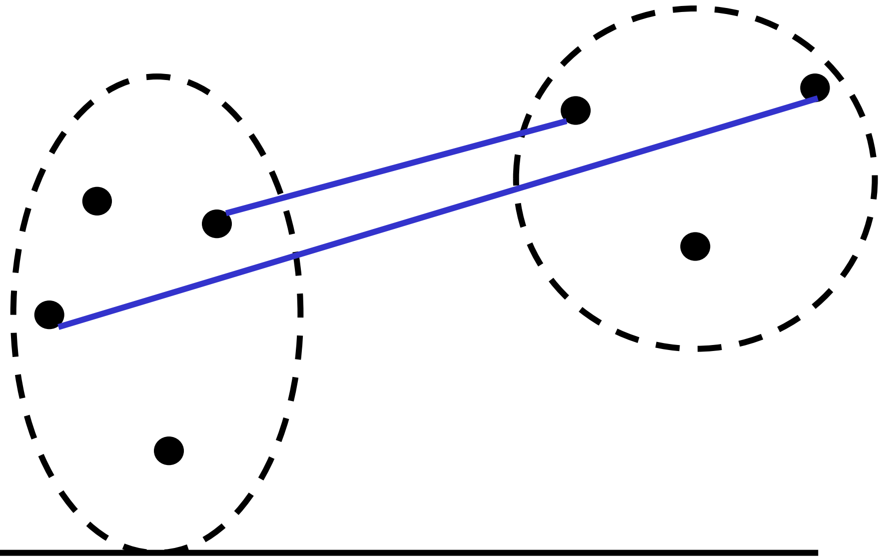


Eigenschaften

- Vorteile
  - Konzeptionell einfach, hübsche, irgendwie intuitive Grafiken
  - Keine Festlegung auf Anzahl Cluster notwendig
- Nachteile
  - Benötigt die Abstandsmatrix als Eingabe
    - $|O|=n$: $O(n^2)$ Platz und $O(n^2)$ Zeit
  - Dazu kommt Clustering selber: $O(n^2*\log(n))$
  - Berechnet keine Cluster per-se – Auswahlschritt notwendig
- Kaum anwendbar für viele (>10.000) Objekte

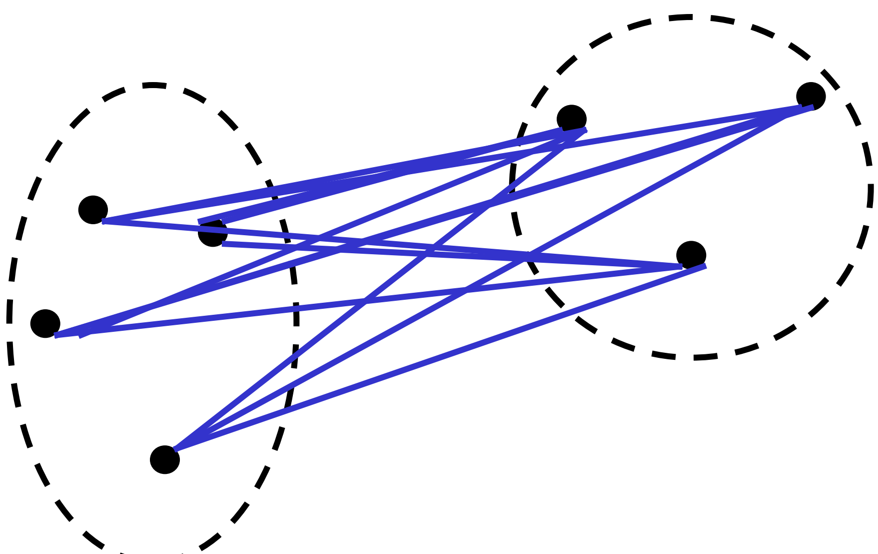
Varianten: Abstandsmaße

- Beim hierarchischen Clustern benötigt man den Abstand zweier Cluster bzw. eines Punktes zu einem Cluster
- Bisher: Clustermittelpunkte verwenden
- Alternativen
  - Single Link: Minimum aller Abstände zwischen zwei Objekten aus je einem Cluster
  - Complete Link: Maximum aller Abstände ...
  - Average Link: Durchschnittlicher Abstand ...
  - Centroid: Abstand der Mittelpunkte



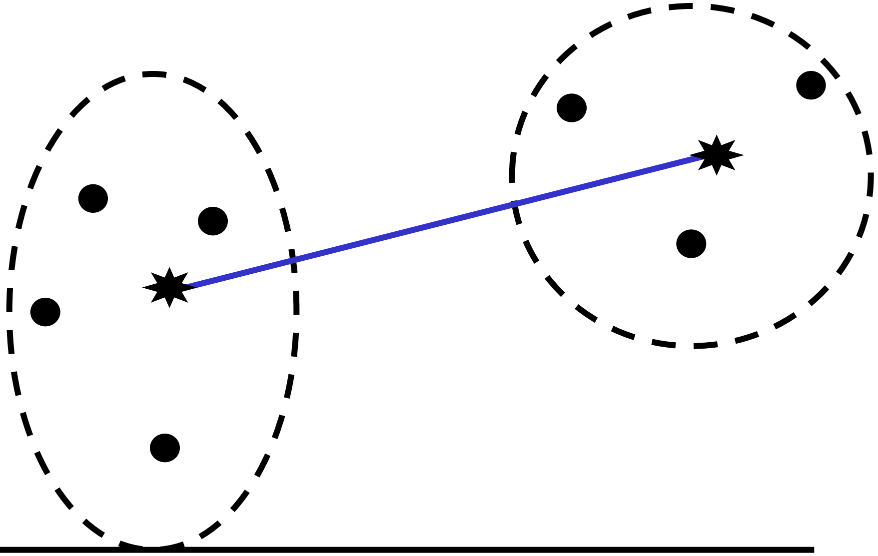
Varianten: Abstandsmaße

- Beim hierarchischen Clustern benötigt man den Abstand zweier Cluster bzw. eines Punktes zu einem Cluster
- Bisher: Clustermittelpunkte verwenden
- Alternativen
  - Single Link: Minimum aller Abstände zwischen zwei Objekten aus je einem Cluster
  - Complete Link: Maximum aller Abstände ...
  - Average Link: Durchschnittlicher Abstand ...
  - Centroid: Abstand der Mittelpunkte

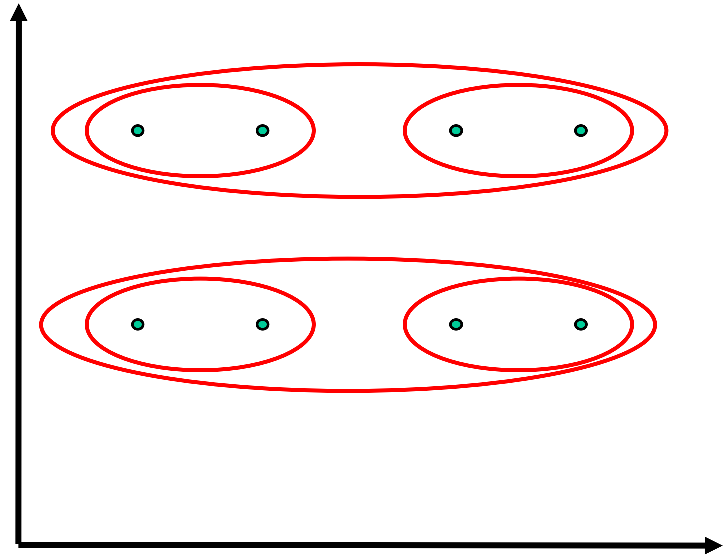


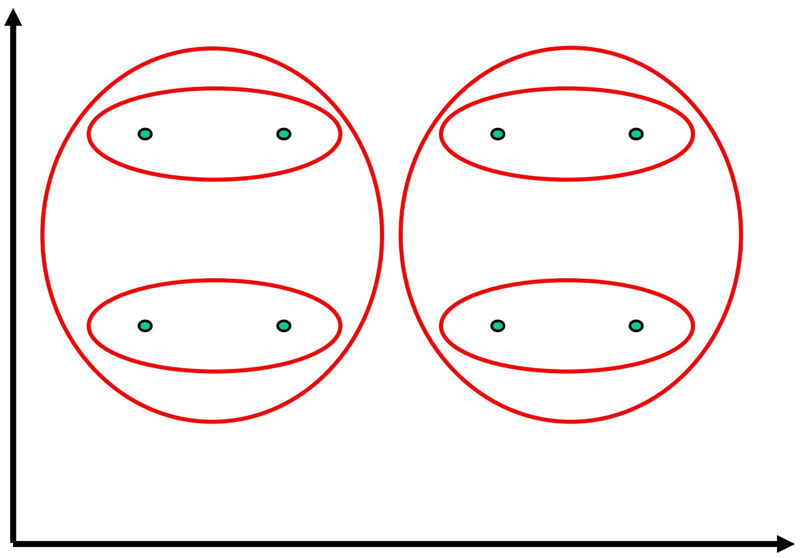
Varianten: Abstandsmaße

- Beim hierarchischen Clustern benötigt man den Abstand zweier Cluster bzw. eines Punktes zu einem Cluster
- Bisher: Clustermittelpunkte verwenden
- Alternativen
  – Single Link: Minimum aller Abstände zwischen zwei Objekten aus je einem Cluster
  – Complete Link: Maximum aller Abstände ...
  – Average Link: Durchschnittlicher Abstand ...
  – Centroid: Abstand der Mittelpunkte



Single-link versus Complete-link





SQL - Distanzmatrix

- Annahmen
  - Alle Objekte und ihre Attribute a, b, … in Tabelle **`objects`**
  - Numerische Attribute
  - Euklidischer Abstand
- Berechnung der Distanzmatrix M?

oid	a	b	c	d	…
1					
2					
3					
4					
5					
6					
…					

```
SELECT  t1.oid, t2.oid,
        sqrt(sqr(t1.a-t2.a)+sqr(t1.b-t2.b)+…)
FROM    objects t1, objects t2
WHERE   t1.oid>t2.oid;
```

SQL – Iteration

- Distanzmatrix materialisieren (teuer)
  - Tabelle **`distance`**
- Anlegen Ergebnistabelle **`cluster(clid,oid1,oid2)`**
- Iterative Berechnung der Cluster
  - Geht nicht mit einer Query
  - Tabelle **`objects`** benötigen wir nicht mehr
  - PL-SQL Programm mit n=|O| Durchläufen
    - Finde Paar $P=(o_1,o_2)$ in **`distance`** mit kleinstem Abstand
      - Schnell mit Index auf Abstandspalte
    - Speichere o_1,o_2 in **`cluster`** mit gemeinsamer clid
    - Füge Abstände von clid zu allen Punkten in **`distance`** ein außer o_1, o_2
    - Löschen alle Tupel in **`distance`**, die ein Objekt aus P beinhalten
      - Schnell mit Indexen auf OID1, OID2

Beispiel

	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							

Distanzmatrix

o1	o2	d
2	1	?
3	1	?
4	1	?
5	1	?
6	1	?
7	1	?
3	2	?
4	2	?
...	...	...

Distanz-
tabelle

o1	o2	d
2	1	?
3	1	?
4	1	?
5	1	?
6	1	?
7	1	?
3	2	?
4	2	?
...	...	...
8	1	?
8	4	?
...	...	...

Sei d(2,3)=min;
Neuer Knoten 8
mit Abständen

o1	o2	d
4	1	?
5	1	?
6	1	?
7	1	?
...	...	...
8	1	?
8	4	?
...	...	...

Einträge mit 2
oder 3 löschen

Berechnung neuer Abstände

Mittelwert der zwei alten Abstände

```
Bestimme $clid, $oldo1, $oldo2;

INSERT INTO distance
SELECT $clid, o.oid1, sum(d.dist)/2
FROM   (SELECT distinct oid1
         FROM distance
         WHERE OID1 not in ($oldo1, $oldo2)) o, distance d
WHERE (d.oid1=o.oid1 and (d.oid2 = $oldo1 or d.oid2=$oldo2))
      or
      (d.oid2=o.oid1 and (d.oid1 = $oldo1 or d.oid1=$oldo2))
GROUP BY o.oid1;
```

Zu diesen Objekten müssen Abstände berechnet werden

Abstände zu Objekt gruppieren

Alte Abstände – Objekte können links oder rechts stehen, selektiert werden immer nur 2 Tupel

Inhalt dieser Vorlesung

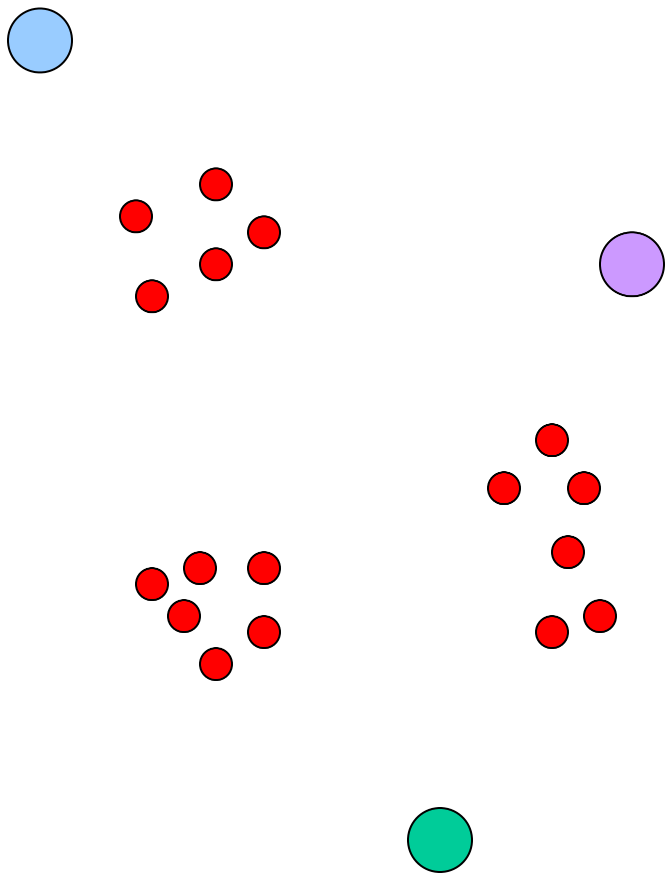
- Einführung
- Hierarchisches Clustering
- Partitionierendes Clustering
  - k-Means
  - k-Medoid und CLARANS
- Dichte-basiertes Clustering
- Weitere Verfahren

K-Means

- Wahrscheinlich bekannteste Clusteringmethode
- Vielen Varianten
- Anzahl k von Clustern ist Eingabeparameter
- Berechnet lokales Optimum bezüglich k-Güte
  - Keine Beachtung des Abstands zu anderen Clustern
- Algorithmus
  - Wähle zufällig k verschiedene Clustermittelpunkte
  - Iteriere
    - Für alle Objekte
      - Berechne Abstand jedes Objekts zu jedem Clustermittelpunkt
      - Weise Objekt seinem nächsten Clustermittelpunkt zu
    - Wenn sich keine Objektzuordnung mehr ändert: STOP
    - Sonst: Berechne neue Clusterzentren

Beispiel 1

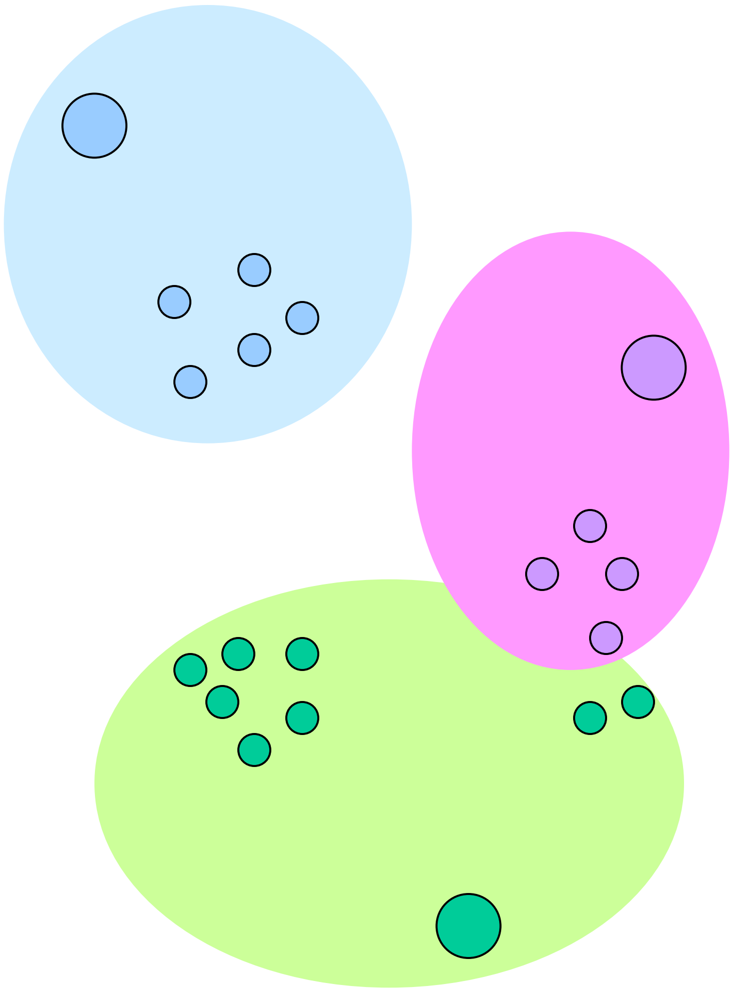
- k=3 zufällige Startwerte auswählen



Quelle: Stanford, CS 262 Computational Genomics

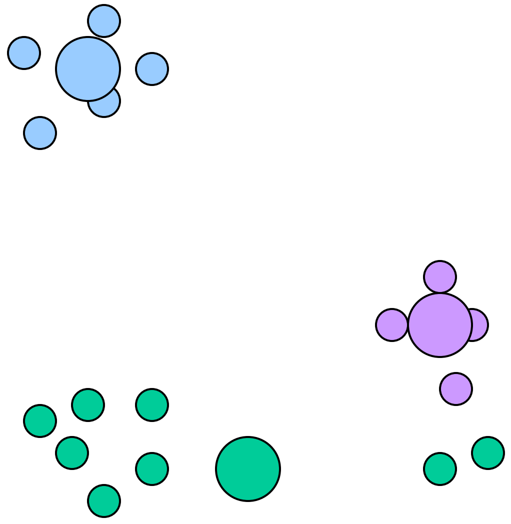
Beispiel 2

- Objekte dem nächsten Clusterzentrum zuordnen



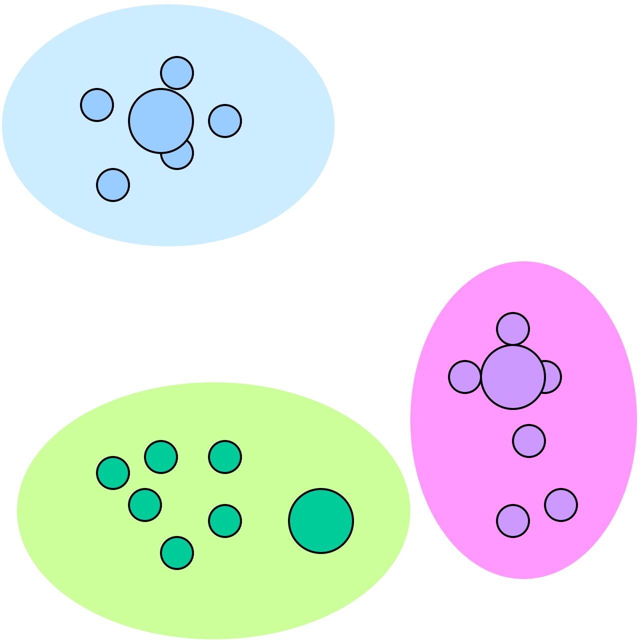
Beispiel 3

- Clustermittelpunkte
  neu berechnen



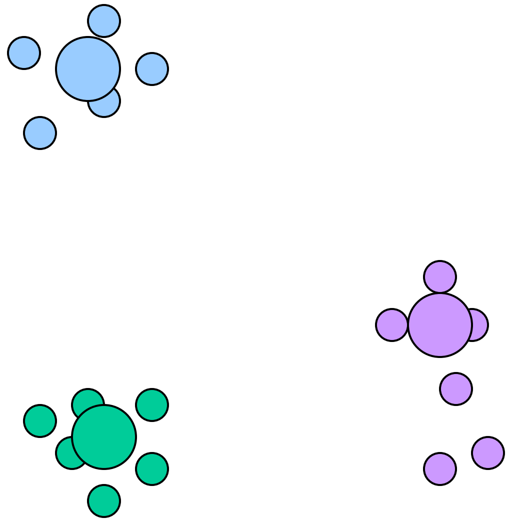
Beispiel 4

- Objekte neu zuordnen



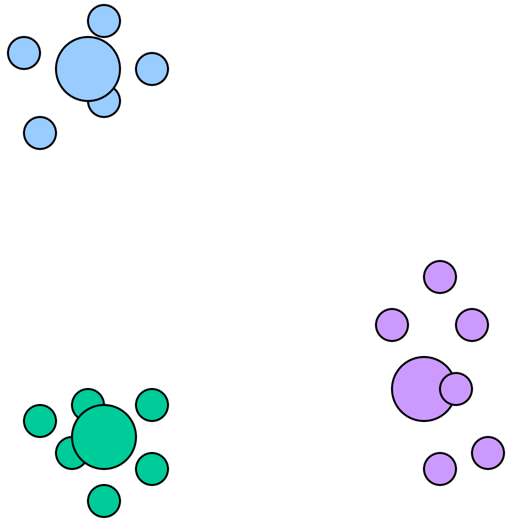
Beispiel 5

- Mittelpunke anpassen



Beispiel 6

- Fertig, keine neuen Zuordnungen mehr



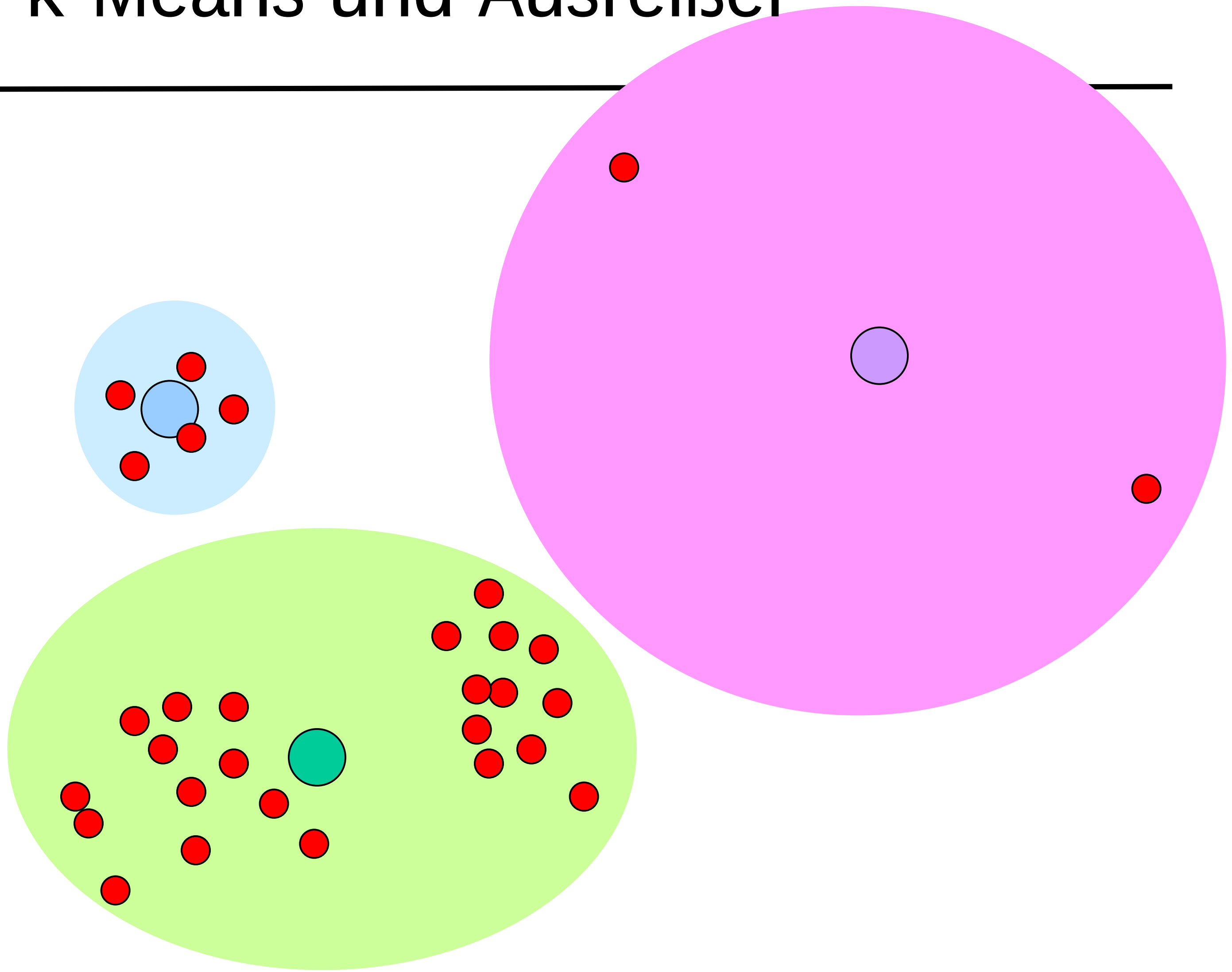
Eigenschaften

- Konvergiert meistens schnell (5-10 Läufe)
- Wenn l die Zahl der Durchläufe ist, brauchen wir
  - Neuzuordnung: n*k Vergleiche Objekte-Zentren
  - Clusterbestimmung: n Vektoradditionen, verteilt auf k Cluster
    - Verwendung des Centroid, nicht des Medoid
  - Zusammen: O(n*k*l)
  - Insbesondere benötigen wir keine vollständige Distanzmatrix
    - Die allermeisten Distanzen werden nie ausgerechnet
- Nachteil: Welches k nehmen wir?
  - Meistens: Verschiedene k probieren
  - Silhouette zur Güteabschätzung verwenden

Varianten

- Wähle initiale Clusterzentren gleichmäßig verteilt im Raum (statt zufälliger Datenpunkte)
  - Schlecht für stark geclusterte Daten, da Mittelpunkte erst einen weiten Weg zurücklegen müssen
- Stop, wenn nur noch wenige (Schwellwert) Objekte ihre Zugehörigkeit geändert haben
  - Schneller, keine (lokal) optimale Lösung mehr
- Starte k-Means mehrmals mit unterschiedlichen Startpunkten und nimm das beste Ergebnis
  - Standardmethode, um zufällig schlechte Startkonstellationen zu verhindern

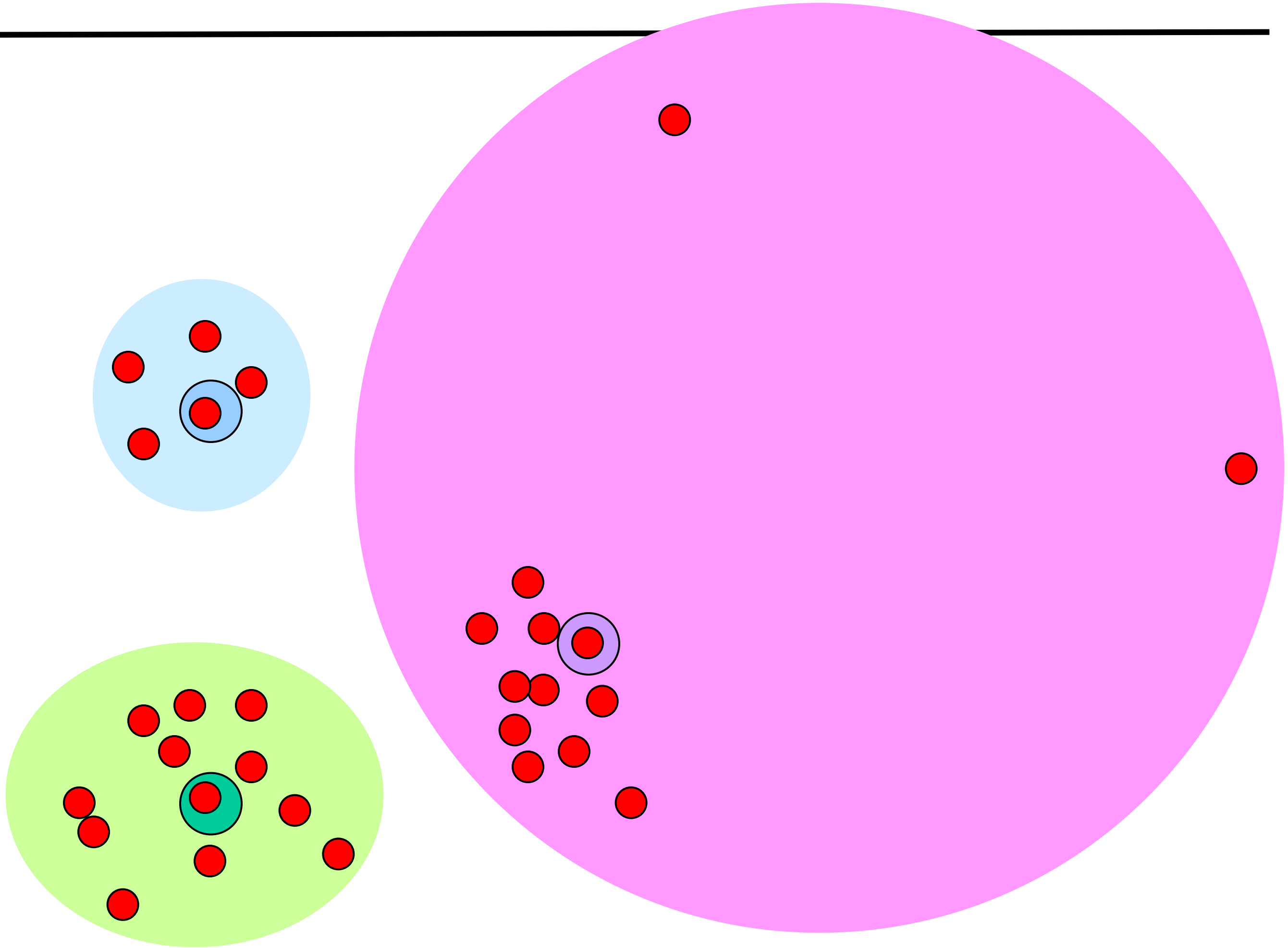
k-Means und Ausreißer



K-Medoid

- K-Medoid: Wahl des mittleren Punktes eines Clusters
- Problem: Berechnung Medoide ist quadratisch in der Clustergröße
- Vorteile
  - Weniger sensitiv bzgl. Ausreißern
  - Funktioniert auch mit kategorialen Werten

k-Medoid und Ausreißer



K-Means in SQL

- **`objects`** mit Objekten und Attributen und Zuordnung
- **`cluster`** mit Koordinaten der Centroide
- Erstes Update: Zuweisung neuer Clusterzentren
- Zweites Update: Berechnung neuer Clustermittelpunkte
- Es fehlt: Berechnung der Clustergröße cn
  – Erfordert weitere Subquery

```
REPEAT
  UPDATE objects o
  SET o.cluster=
    (SELECT cid
     FROM (SELECT dist(o,c) d
           FROM cluster c
           ORDER BY d)
     WHERE ROWNUM=1);
  IF %SQLCOUNT% != 0
    UPDATE cluster
    SET (a,b,…)=
      (SELECT sum(a)/cn,sum(b)/cn, …
       FROM objects o
       WHERE o.cluster=cid
       GROUP BY o.cluster);

  ELSE
    BREAK;
  ENDIF;
UNTIL FALSE;
```

CLARANS [NH94]

- Beobachtung bei k Mediod (oder K Means)
  - Die meisten Objekte (A) finden „ihre“ Cluster sehr schnell und wechseln dann nie mehr
  - Wenige Objekte (B) brauchen länger zur Konvergenz und lassen den Algorithmus lange laufen
  - Dabei müssen Objekte aus A immer wieder mit allen Clustern verglichen werden – ohne dass sich etwas ändert
- Idee von CLARANS: Randomisierung
  - Vergleiche immer nur wenige (z.B. 10%) zufällig gewählte Objekte mit einem zufällig gewählten Cluster
  - Setze gewähltes Objekt als Mediod, wenn bessere Lösung
  - Kann weit vom Optimum entfernte Lösung generieren – wiederhole häufig (>100 Mal)

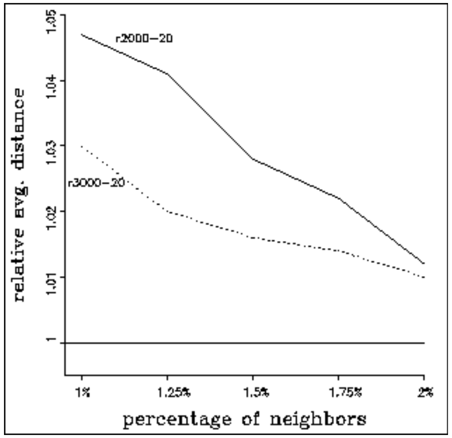
Code

- Idee: Teste nur manche Paare
- maxneighbor viele
- dafür starte öfter (maxtest)
- TD: Total distance (alle Objekte zu ihren Medoiden)
  - Neuberechnung notwendig

```
TD_best := maxint;         // Bester Gesamtabstand
C_best := ∅;               // Beste Medoidmenge
O;                         // Alle Objekte
for r = 1 … maxtest do
  C := {wähle zufällig k Objekte als Medoide};
  O := O \ C;
  weise Objekte nächstem Medoid zu;
  berechne TD;
  i := 0;
  for i := 1 … maxneighbor do
    Wähle zufällig m∈C, n∈O;
    if TD_N↔M < TD then      // Diese tauschen?
      O := O ∪ m \ n;
      C := C ∪ n \ m;
      TD := TD_N↔M;
    end if;
  end for;
  if TD < TD_best then     // Neues Optimimum?
    TD_best := TD;
    C_best := C;
  end if;
end do;
return TD_best, C_best;
```

Vergleich [ES00]

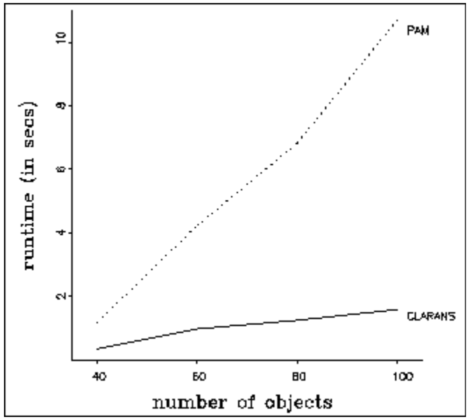
Qualität



TD(CLARANS)

TD(PAM)

Laufzeit



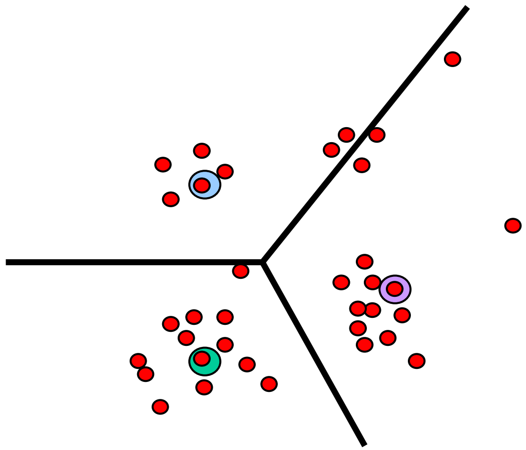
Quelle: [ES00]

- Unwesentlich schlechtere Ergebnisse (1-5%)
- Viel bessere Laufzeit (nahezu linear)
- Nicht untypisch: Wenn die Daten „gut" clustern, dann findet man diese Cluster sehr schnell
  - Optimale Zuordnung der wenigen problematischen Objekte benötigt viel Zeit, bringt aber nur wenig Verbesserung

Inhalt dieser Vorlesung

- Einführung
- Hierarchisches Clustering
- Partitionierendes Clustering
- Dichte-basiertes Clustering
- Weitere Verfahren

Aber ...



Quelle: [FPPS96]

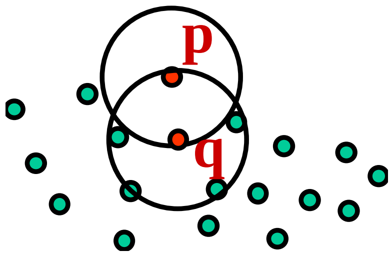
- K-Means (und CLARANS und k-Medoid und viele andere) finden nur konvexe Cluster
  - Das ergibt sich aus der Nähe zu einem Mittelpunkt
- Anderes Kriterium: Nähe zu genügend vielen anderen Punkten im Cluster

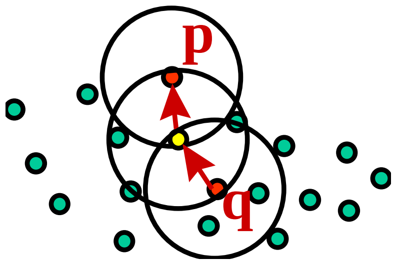
Dichtebasiertes Clustering [EKSX96]

- Sucht nach Regionen hoher Dichte
  - Anzahl Cluster ist nicht vorbestimmt
  - Form ist beliebig (auch geschachtelte Cluster sind möglich)
- Bekanntester Vertreter: DBSCAN
- Wie definiert man „dichte“ Bereiche?
  - Jeder Punkt eines Clusters hat genügend viele nahe Nachbarn
    - Genügend: Parameter; Nachbar: Parameter
  - Alle Punkte eines Clusters sind über nahe Nachbarn voneinander erreichbar

Grundbegriffe

- Definition
  Geg. Parameter ε („Nachbar“) und minpts („viele“). Sei $N\varepsilon(o)$ die ε-Nachbarschaft von Punkt o.
  - *Ein Objekt o heißt Kernobjekt, wenn $|N\varepsilon(o)| \geq minpts$*
  - *Ein Objekt p ist direkt dichte-erreichbar von einem Objekt q, wenn q ein Kernobjekt ist und $p \in N\varepsilon(q)$*
    - *p muss kein Kernobjekt sein (Rand)*
  - *p ist dichte-erreichbar von q, wenn es eine Kette von direkt dichte-erreichbaren Objekten zwischen p und q gibt.*

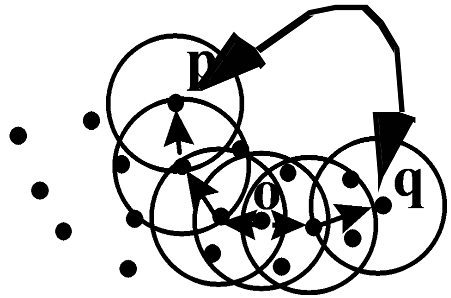




- Bemerkung
  - Dichte-Erreichbarkeit erzeugt einen Kernbereich und einen Rand

Weitere Grundbegriffe

- Definition (Voraussetzungen wie eben)
  - *Zwei Objekte p und q sind dichte-verbunden, wenn es mindestens ein Kernobjekt o gibt, von dem p und q dichte-erreichbar sind.*
    - *Auch Randpunkt sind also dichte-verbunden*



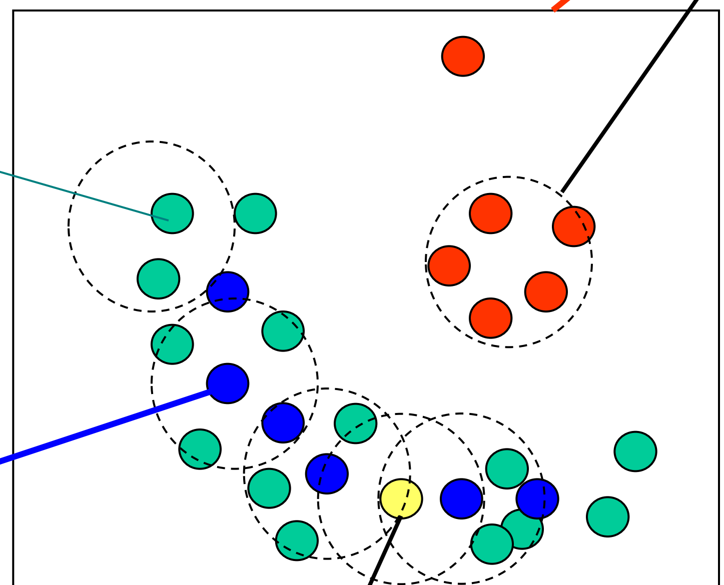
  - *Ein Cluster ist eine Teilmenge $C \subseteq O$ für die gilt*
    - *Maximalität: $\forall\, p,q \in O$: wenn $p \in C$ und q dichte-erreichbar von p ist, dann ist auch $q \in C$*
    - *Verbundenheit: $\forall\, p,q \in C$: p ist dichte-verbunden mit q*
  - *Ein Clustering von O ist die Menge aller Cluster von O, die mindestens ein Kernobjekt enthalten*
  - *Alle Punkte, die nicht in einem Cluster sind, heißen Rauschen*
- Bemerkung
  - Es gilt: Wenn $p \in C$ ein Kernobjekt: $C = \{o \in O \mid o$ dichte-erreichbar von $p\}$
  - Cluster sind nicht notwendigerweise disjunkt
    - Können im Rand überlappen

Beispiel

Rauschen
(Kein Rauschen, wenn minpts=4)

Randpunkte

Kernpunkte



ε = ...

minpts = 5

Randpunkt befindet sich in zwei Clustern

Algorithmus

- Aus der Definition ergibt sich unmittelbar ein Algorithmus zum Finden des dichtebasierten Clusterings einer Objektmenge O
  - Das Clustering ist eindeutig

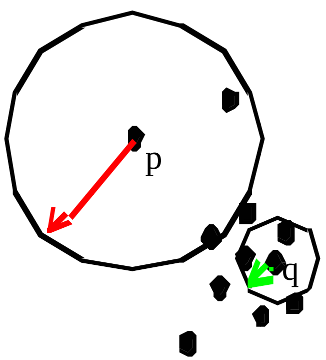
```
clusterCount := 1;
for i from 1 to |O| do                       // Alle Punkte ansehen
  o := O.get(i);
  if o.clusterID = NULL and                  // Punkt in keinem Cluster
     kernobjekt(o) then                      // … und ein Kernobjekt
                                             // … also Kern eines neuen C
       expandCluster(  o, O);                // Ganzen Cluster rekursiv
                                             // berechnen (und Kernobjekte
                                             // aus O entfernen)
       clusterCount++;                       // Nächster Cluster
```

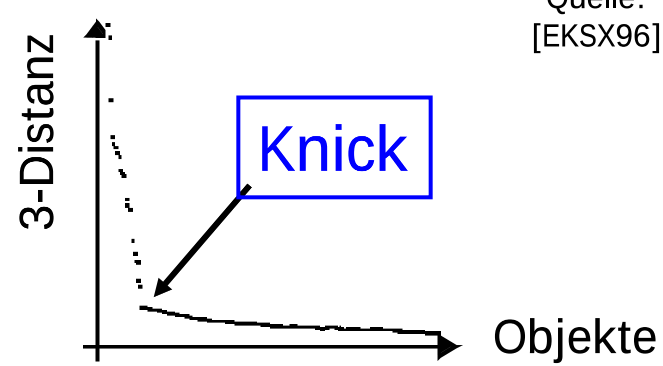
Analyse

- Benötigt – oberflächlich gesehen – nur einen Lauf durch die Daten
- Aber: expandCluster ist teuer
  - Sucht rekursiv nach allen Punkten in ε-Nachbarschaften aller dichte-erreichbaren Punkte
  - Ohne multidimensionalen Index
    - Alle paarweisen Distanzen vorberechnen (Distanzmatrix – teuer)
    - Bei Anfrage für o: Alle Objekte p verwerfen mit $p \notin N_\varepsilon(o)$
    - Benötigt $O(n^2)$ Zeit und Platz - schlecht
  - Mit multidimensionalem Index
    - MDI muss Nachbarschaftsanfragen unterstützen
    - Damit: O(n* Aufwand für eine ε-Query)
      - Laufzeit hängt dann idR von ε und Dimensionalität der Daten ab
- Gleiches Problem beim Test **`kernobjekt()`**

Wie findet man die Parameter?

- Idee: Finde erst den am wenigsten dichten Cluster in den Daten
  – Der aber trotzdem ein Cluster ist – Definitionssache
- Für ein Objekt o ist seine k-Distanz die Entfernung des k-nächsten Objekt
- Damit können wir ein k-Distanz-Diagramm bauen
- Wähle den „Knick“ in der Verteilung

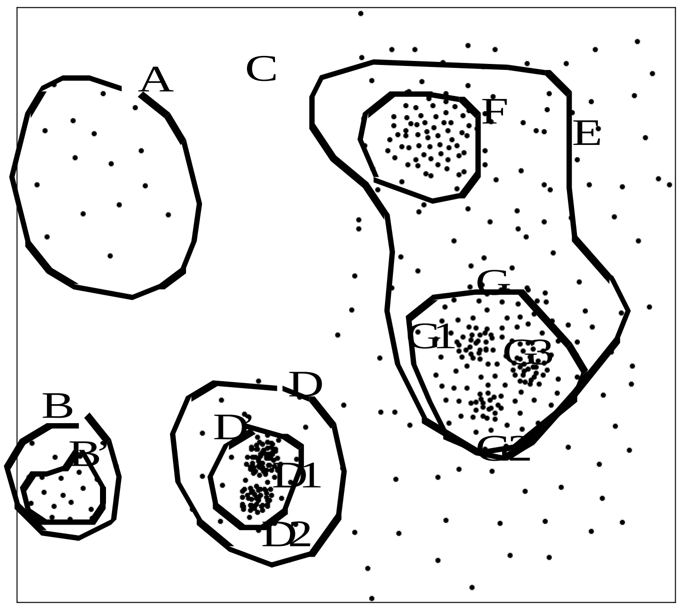


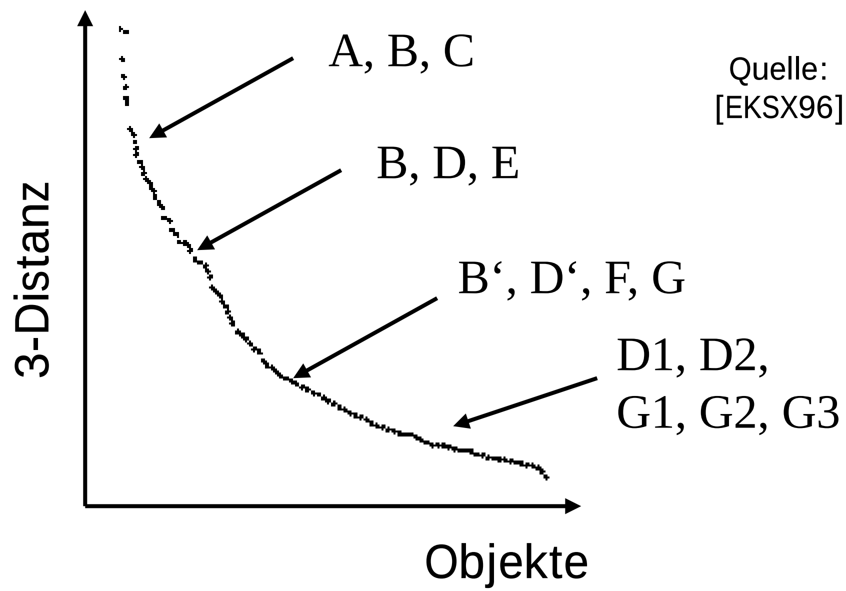


Quelle: [EKSX96]

Wenn es keinen Knick gibt?

- Stark unterschiedliche Dichte in verschiedenen Bereichen des Raumes
  - Viele (kleine) Knicks
  - Mit einem Parameterpaar minpts, ε kann man das nicht beschreiben



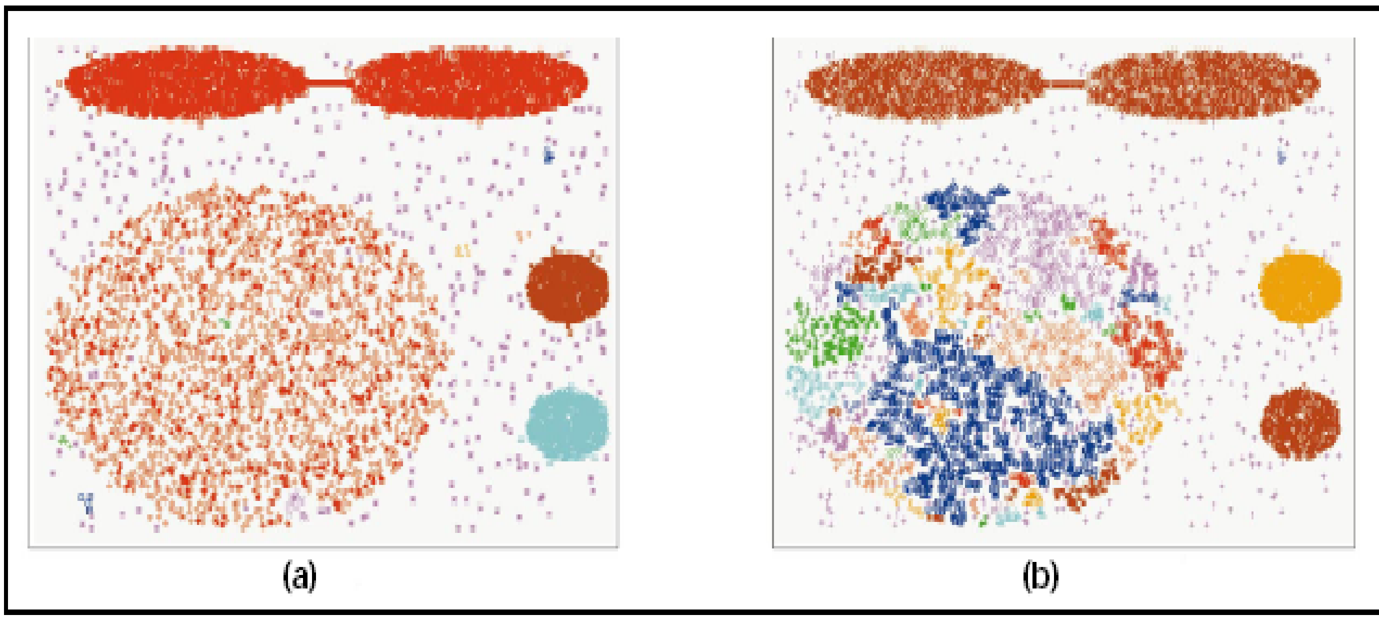


Quelle: [EKSX96]

Sensitivität

- Wählen des „richtigen“ ε ist essentiell

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.



Inhalt dieser Vorlesung

- Einführung
- Hierarchisches Clustering
- Partitionierendes Clustering
- Dichte-basiertes Clustering
- Weitere Verfahren

Min-k-Cut Clustering

- Clustering in graph-theoretic concepts
- Definition
  Let $G=(V,E)$ be a complete, weighted, undirected graph with $V=O$ and $w(o_1,o_2) = d(o_1, o_2)$.
  - *A k-cut of G is a set S of edges such $G'=(V,E\backslash S)$ has exactly k connected components.*
  - *A min-k-cut of G is a k-cut of G such that w(S) is maximal*
- Notes
  - Every k-cut is a clustering of G into k clusters
  - Optimizes for the distance to other clusters
  - Finding a min-k-cut is in O(|V|^k^2)
  - Not feasible in practice

Girvan–Newman algorithm [GM02]

- Popular method for community detection in social networks
- Idea: Remove edges that are important for connecting nodes
  - An edge is important if it is contained in many shortest paths between any pair of nodes
  - Removing important edges will destroy connections – clustering
- Algorithm
  - Compute "betweenness score" of each edge (Floyd-Warshall)
  - Remove edge with the highest betweenness value
  - Re-compute betweenness values (can be done an Delta-Set only)
  - Iterate until all nodes are separate clusters
- Remarks
  - A decisive hierarchical clustering method
  - Focus on edges, not on nodes
  - Very slow – computing betweenness is $O(n^3)$ for dense graphs

Literatur

- Ester, M. and Sander, J. (2000). "Knowledge Discovery in Databases". Berlin, Springer.
- Han, J. and Kamber, M. (2006). "Data Mining. Concepts and Techniques", Morgan Kaufmann.
- Ester, M., Kriegel, H. P., Sander, J. and Xu, X. (1996). "A density-based algorithm for discovering clusters in large spatial databases". Conference on Knowledge Discovery in Databases.
- Ng, R. T. and Han, J. (1994). "Efficient and Effective Clustering Methods for Spatial Data Mining". Int. Conf. on Very Large Databases, Santiago, Chile.
- Girvan M. and Newman M. E. J. (2002). “Community structure in social and biological networks”, Proc. Natl. Acad. Sci. USA 99

Selbsttest

- Welche Cluster-Verfahren gibt es?
- Definieren Sie das formale Optimierungsproblem zum Clustern der Menge O für eine gegebene Clusterzahl k
- Welche Komplexität hat hierarchisches Clustering? Begründen Sie ihr Angabe.
- Welche Eigenschaften hat k-Means Clustering im Vergleich zu hierarchischem Clustering?
- Warum ist k-Means anfällig für Ausreisser?
- Welche Variante des hierarch. Clusterns ist weniger anfällig für Ausreisser: Single, complete oder average?
- Clustern Sie die folgende Objektmenge hierarchisch nach (a) der Centroidmethode und (b) nach Single-Linkage