Non-Standard-Datenbanken und Data Mining

Clusterbildung / Häufungsanalyse

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Übersicht

- Semistrukturierte Datenbanken (JSON, XML) und Volltextsuche
- Information Retrieval
- Mehrdimensionale Indexstrukturen
- Cluster-Bildung
- Einbettungstechniken
- First-n-, Top-k-, und Skyline-Anfragen
- Probabilistische Datenbanken, Anfragebeantwortung, Top-k-Anfragen und Open-World-Annahme
- Probabilistische Modellierung, Bayes-Netze, Anfragebeantwortungsalgorithmen, Lernverfahren,
- Temporale Datenbanken und das relationale Modell,
- Probabilistische Temporale Datenbanken
- SQL: neue Entwicklungen (z.B. JSON-Strukturen und Arrays), Zeitreihen (z.B. TimeScaleDB)
- Stromdatenbanken, Prinzipien der Fenster-orientierten inkrementellen Verarbeitung
- Approximationstechniken für Stromdatenverarbeitung, Stream-Mining
- Probabilistische raum-zeitliche Datenbanken und Stromdatenverarbeitungsssysteme: Anfragen und Indexstrukturen, Raum-zeitliches Data Mining, Probabilistische Skylines
- Von NoSQL- zu NewSQL-Datenbanken, CAP-Theorem, Blockchain-Datenbanken

Data Mining: Clusterbildung

- Form des unüberwachten Lernens
- Suche nach natürlichen Gruppierungen von Objekten
  - Klassen direkt aus Daten bestimmen
    - Hohe Intra-Klassen-Ähnlichkeit
    - Kleine Inter-Klassen-Ähnlichkeit
  - Ggs.: Regression / Klassifikation
- Distanzmaße: erster Ansatz
  - 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

Wenn p $\longrightarrow \infty$: Max
Wenn p $\longrightarrow -\infty$: Min

Grundsätzliche Vorgehensweisen

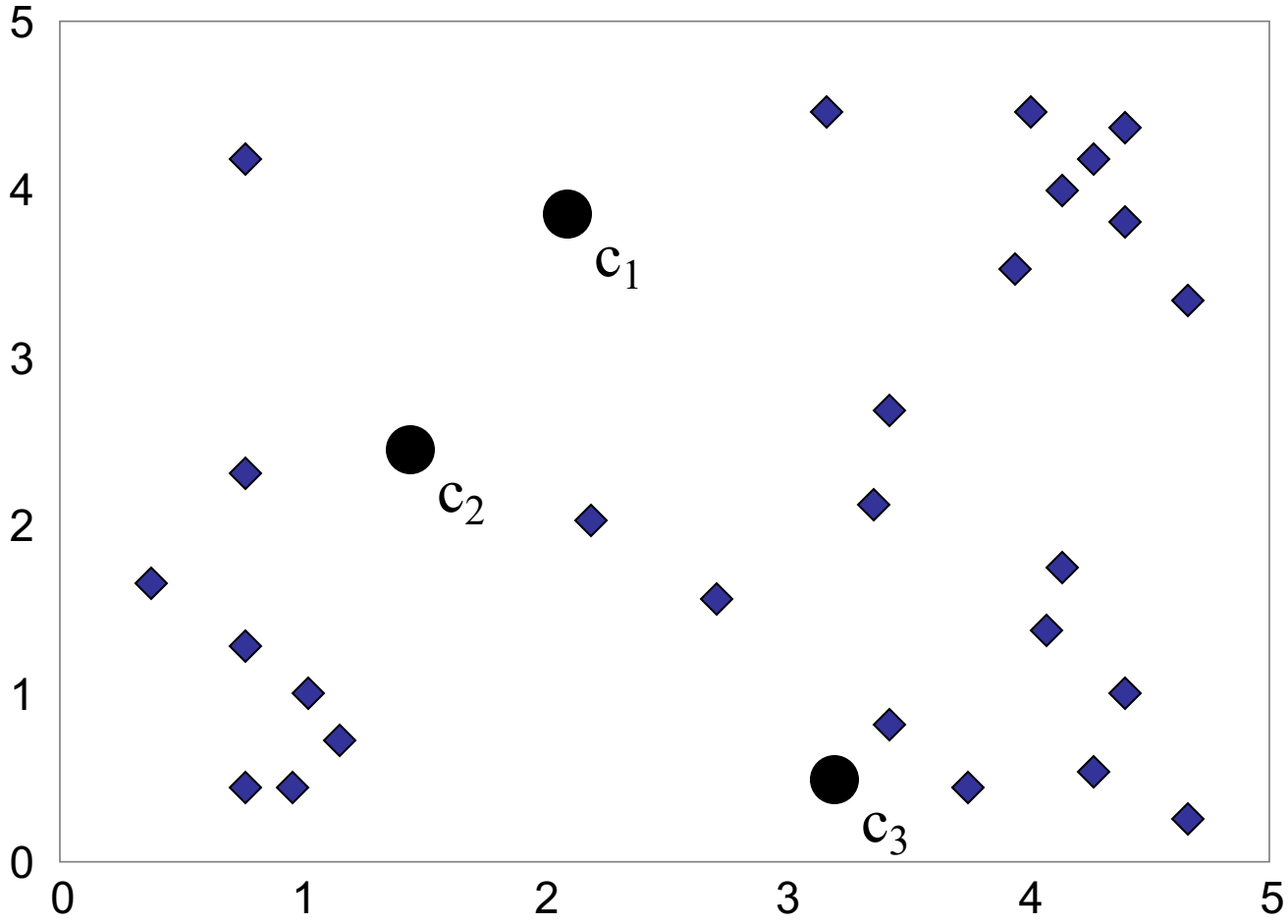
- Partitionierungsbasiert
  - K-Means

    S. P. Lloyd: Least square quantization in PCM. In: Bell Telephone Laboratories Paper. **1957**, später erst in einer Zeitschrift erschienen: S. P. Lloyd: Least squares quantization in PCM. In: IEEE Transactions on Information Theory. 2. Auflage. Band 28, S. 129–137, **1982**

  - K-Medoids
  - BFR
  - DBSCAN
- Hierarchisch (zusammenballende Clusterbildung)
  - CURE
  - BIRCH
  - Spektrale Clusterbildung
- Multidimensionale Indexierungstechniken notwendig

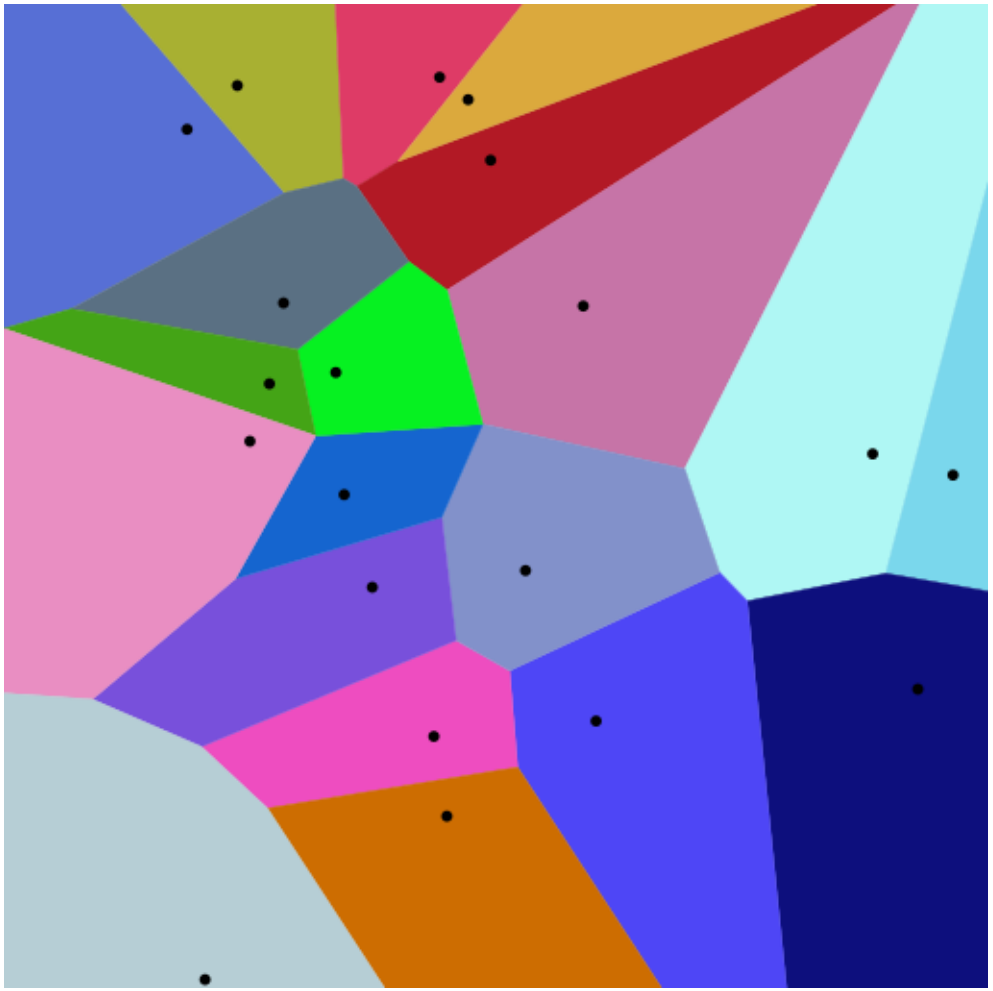
K-means Clusterbildung: Initiale Zentroiden

Distanzmaß: Euklidische Distanz



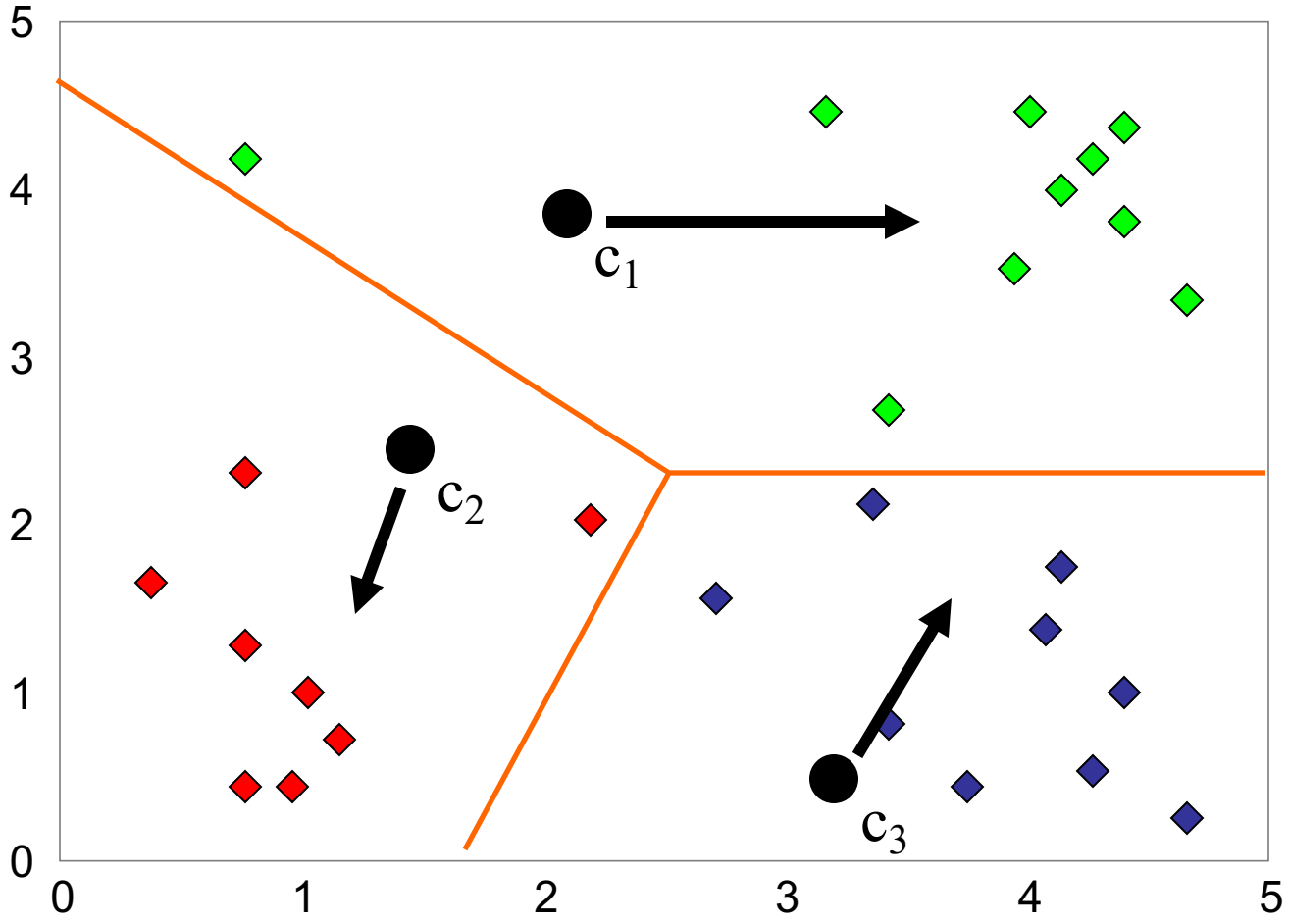
$$C_i^t = \left\{ x_j : \left\| x_j - c_i^t \right\|_2 \leq \left\| x_j - c_r^t \right\|_2 \text{ for all } r = 1 \ldots k, r \neq i \right\}$$

Voronoi-Diagramm



K-means Clusterbildung: Voronoi-Diagram

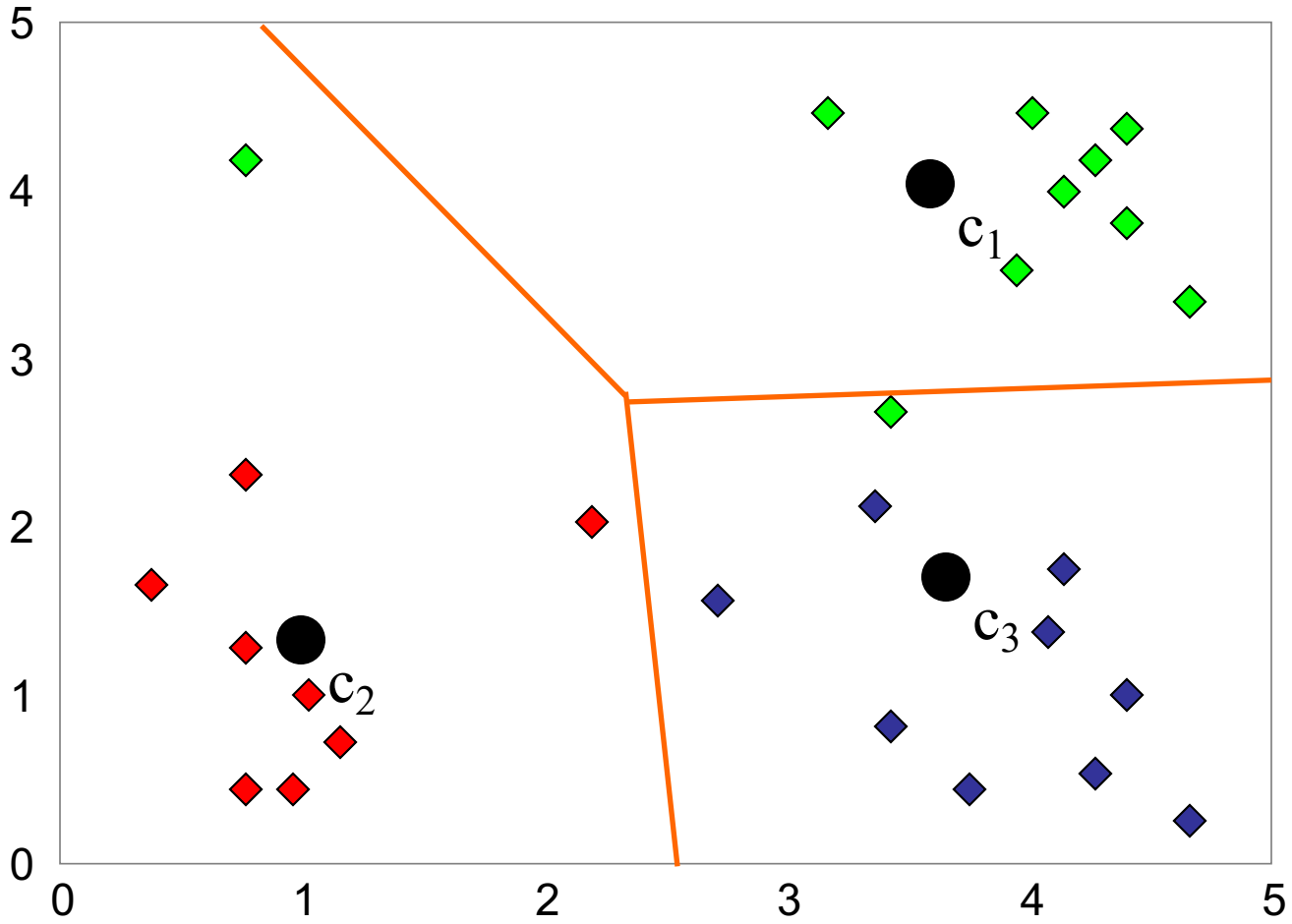
Distanzmaß: Euklidische Distanz



$$c_i^{t+1} = \frac{1}{|C_i^t|} \sum_{x_j \in C_i^t} x_j$$

K-means Clusterbildung

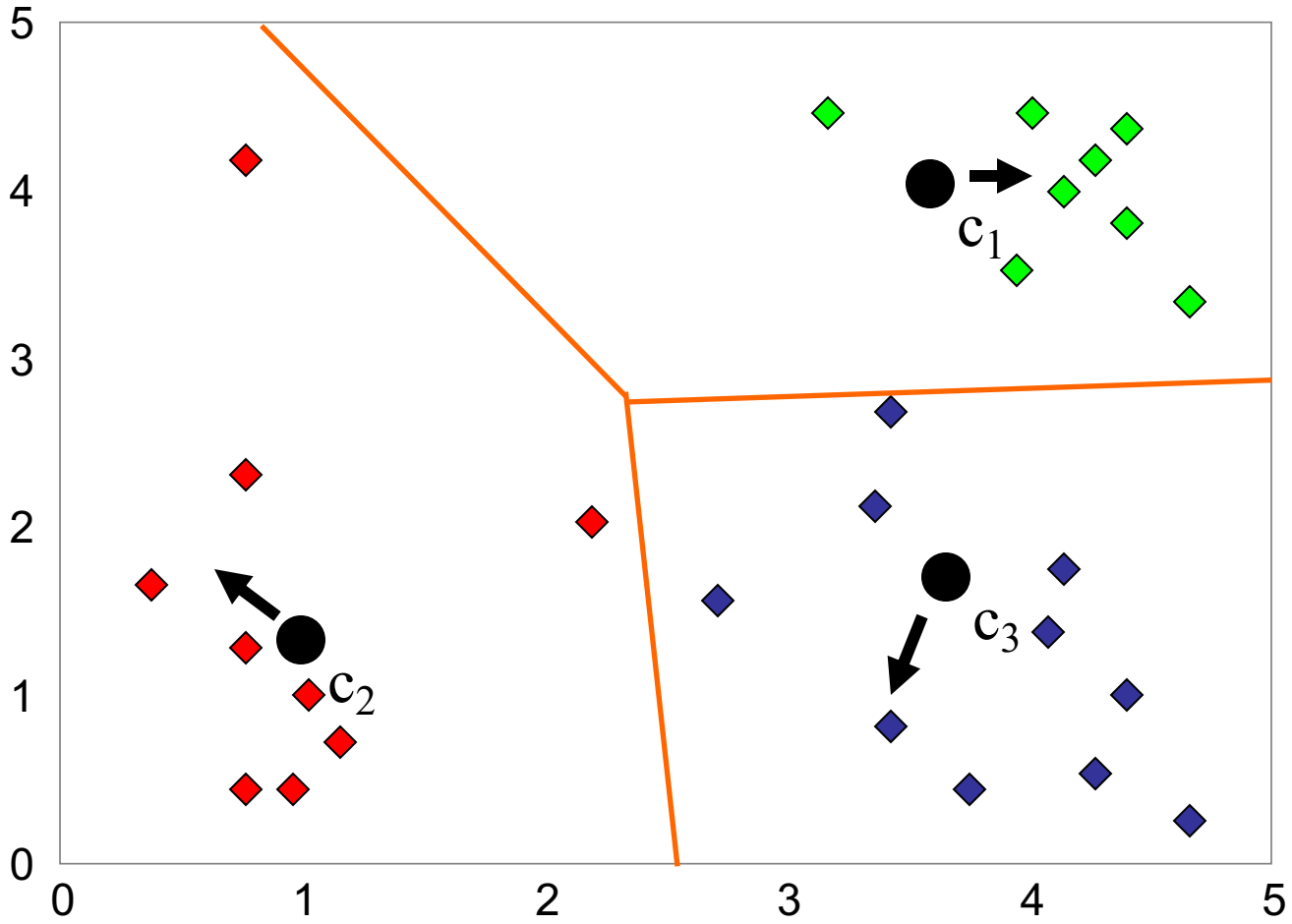
Distanzmaß: Euklidische Distanz



J. B. MacQueen: "Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*", Berkeley, University of California Press, 1:281-297, **1967**

K-means Clusterbildung: Voronoi-Iteration

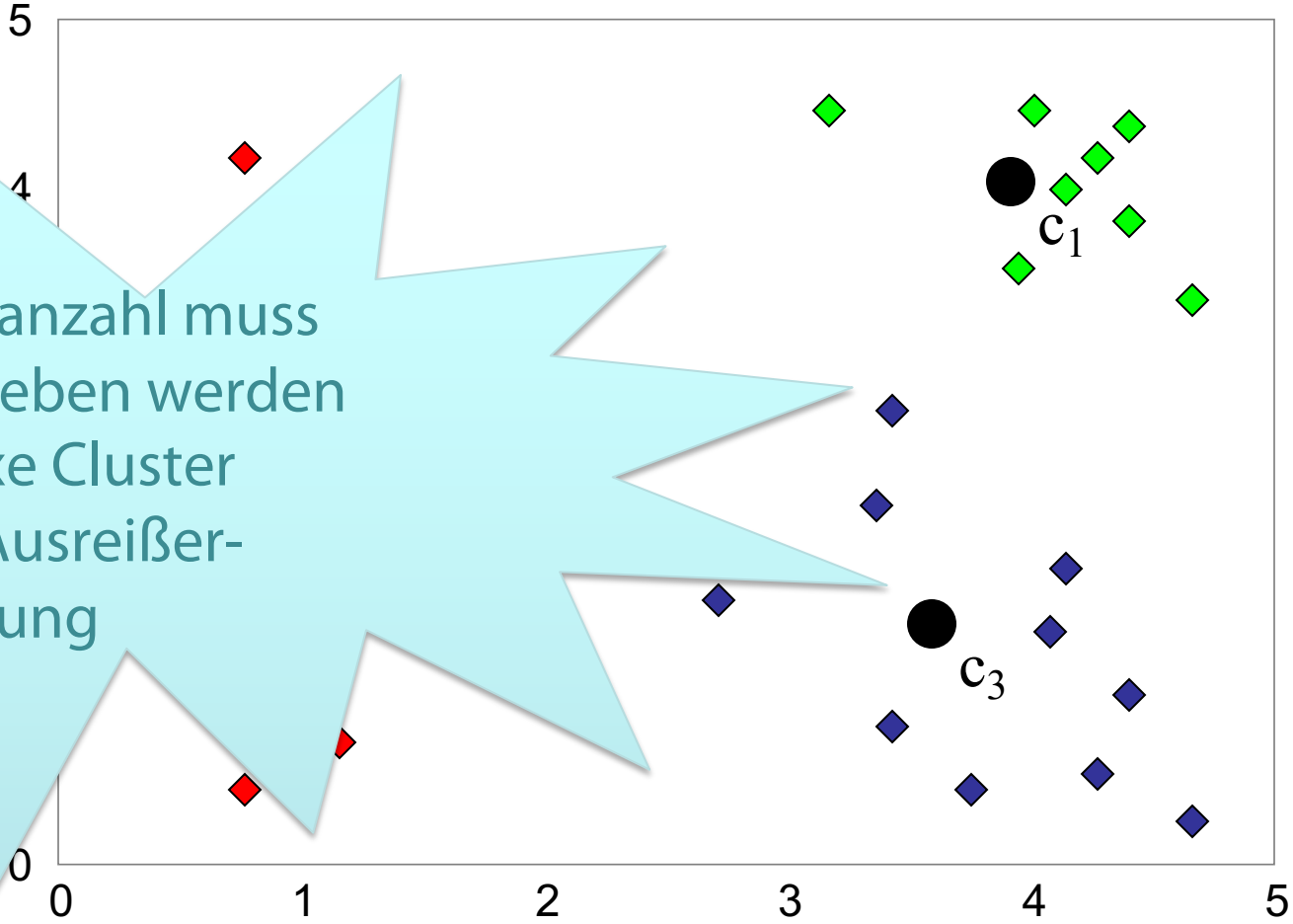
Distanzmaß: Euklidische Distanz



J. B. MacQueen: "Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*", Berkeley, University of California Press, 1:281-297, **1967**

K-means Clusterbildung

Distanzmaß: Euklidische Distanz



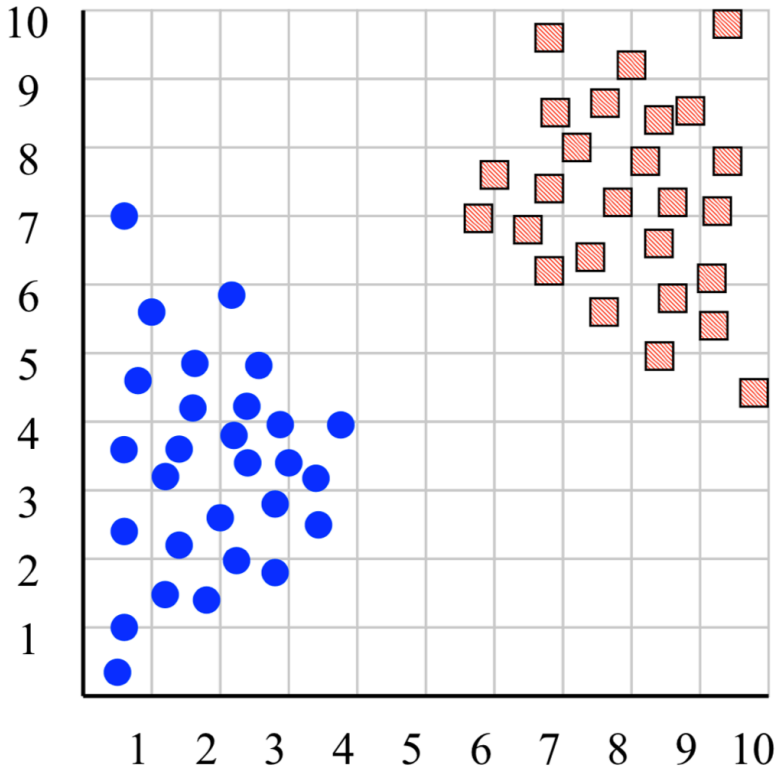
- Clusteranzahl muss vorgegeben werden
- Konvexe Cluster
- Keine Ausreißer-Erkennung

J. B. MacQueen: "Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*", Berkeley, University of California Press, 1:281-297, **1967**

Bestimmung von k

- Was ist die *richtige* Anzahl von Clustern?
- Schlecht gestelltes Problem
- Betrachten wir ein Beispiel (Grashüpfer/Heuschrecken-Datensatz)

Annahme:
Clusteranzahl nicht bekannt



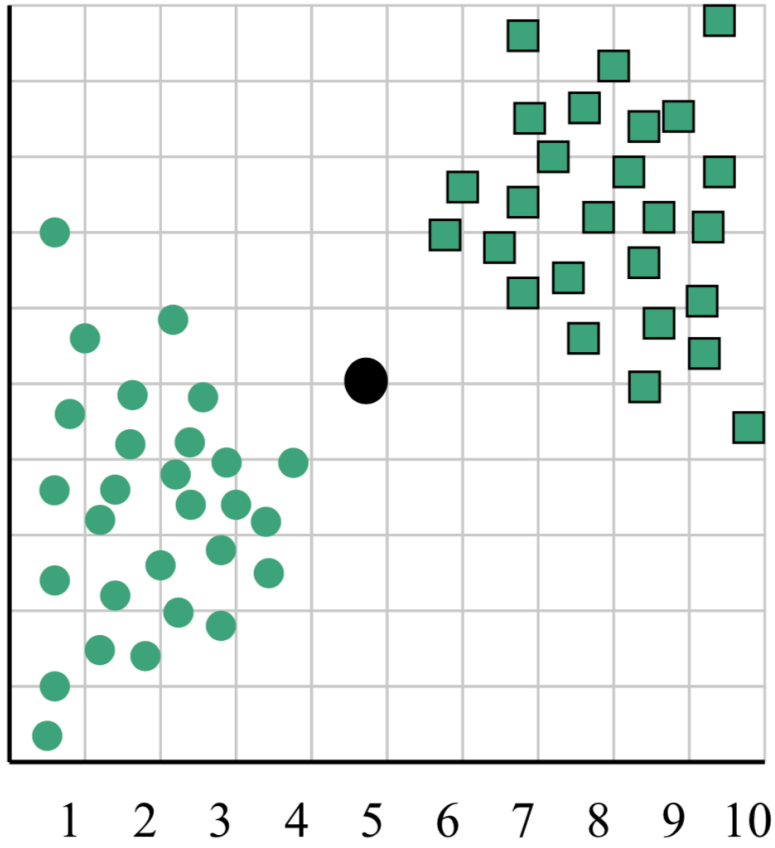
Bestimmung von k

Sei k = 1: $se_K =$ 873.0

$$se_{K_i} = \sum_{j=1}^{m_i} \| t_{ij} - C_i \|_2$$

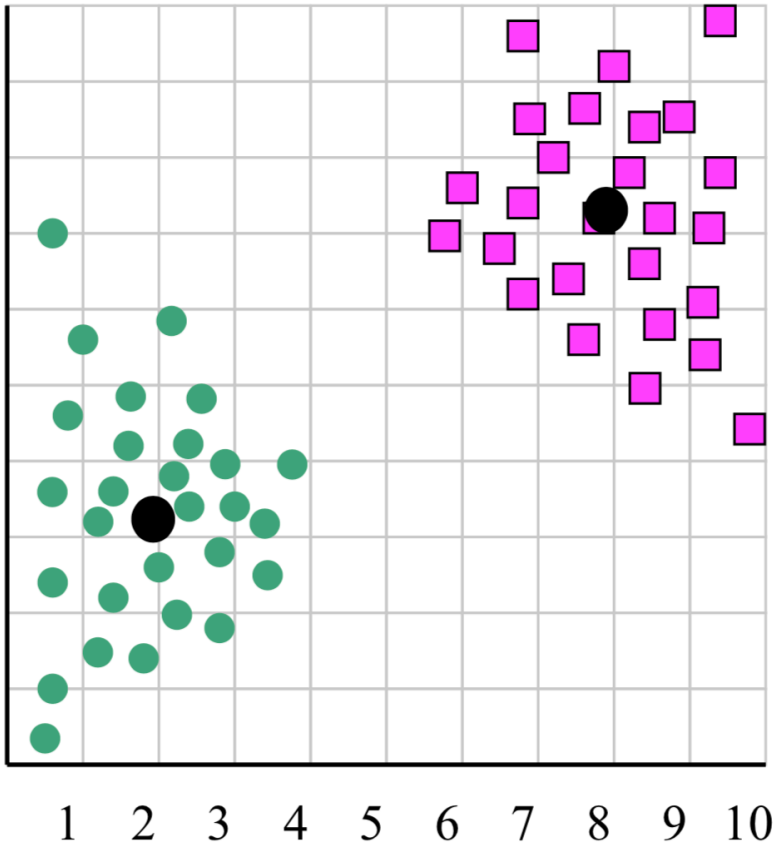
$$se_K = \sum_{j=1}^{k} se_{K_j}$$

se = squared error



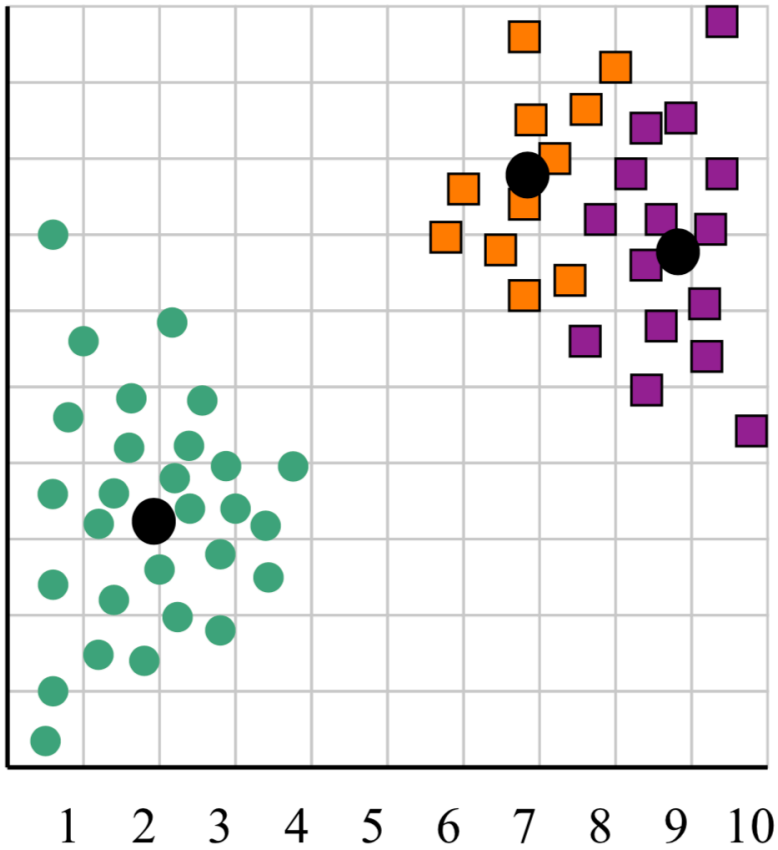
Bestimmung von k

Sei k = 2: $se_K =$ 173.1



Bestimmung von k

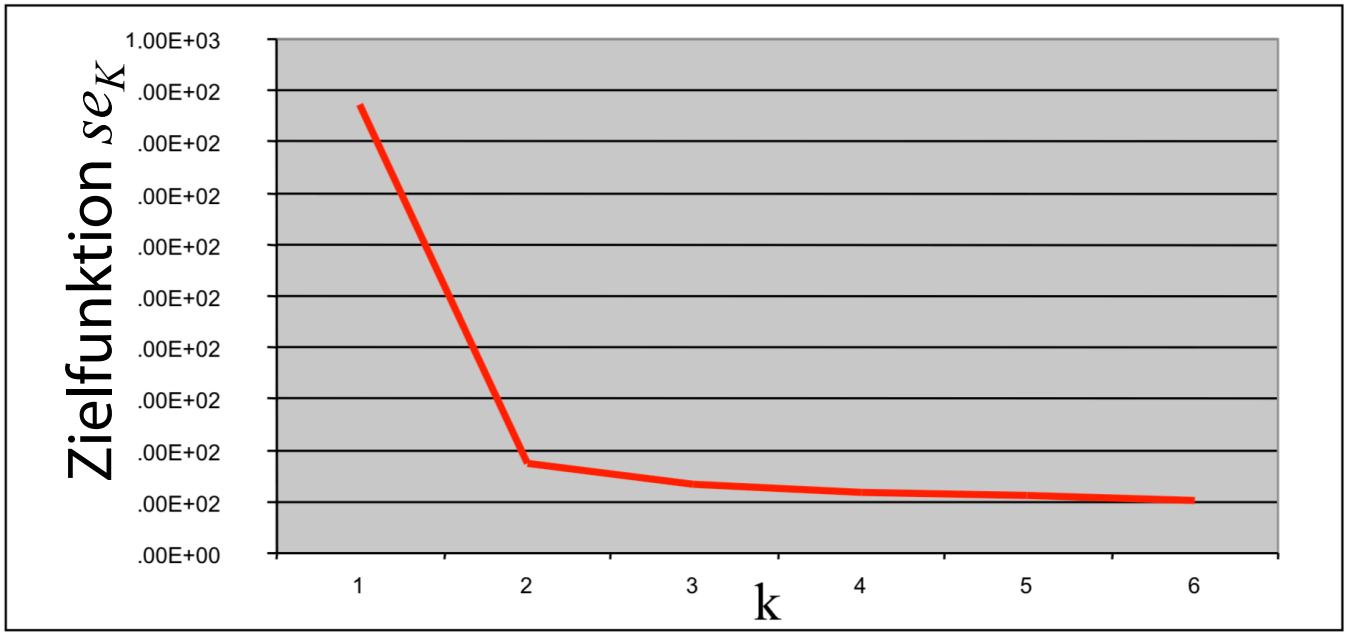
Sei k = 2: $se_K =$ 133.6



Bestimmung von k

Darstellung der Werte als Graph

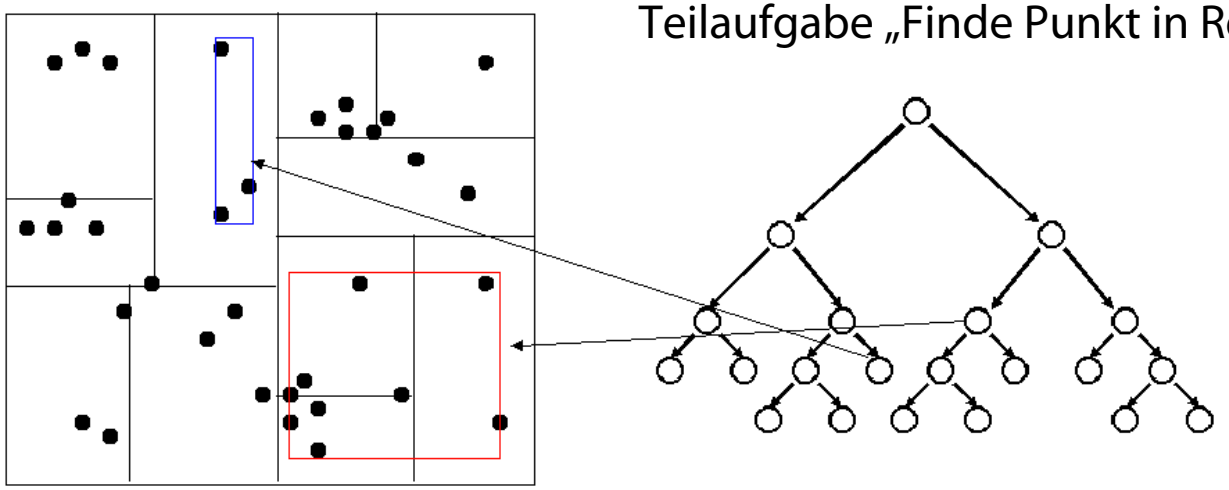
- Für k=2 gibt es eine abrupte Änderung
- Kniefindung oder Ellenbogenfindung



Bestimmung der Startwerte

- Was sind gute initiale Cluster?
- Alternativen zu ihrer Bestimmung:
  - Nehme *k* Zufallspunkte
  - Nehme kleine Zufallsstichprobe und bilde *k* Cluster
  - Nehme Zufallsstichprobe, wähle zufälligen Punkt, und dann wähle *k–1* weitere Punkte, jeden soweit entfernt von den vorigen wie möglich

k-d-Bäume zur Effizienzsteigerung



Teilaufgabe „Finde Punkt in Region“

K-Means-Clusterbildung (n = Anzahl der Datenpunkte, k = Anzahl Cluster)

- Zeitkomplexität Aufbau: $O(dn \cdot \log n)$ (Sortierung nach d Dimensionen)
- Zeitkomplexität k Regionenanfragen: $\Omega(k \cdot \log n)$ statt $O(k \cdot n)$ naiv
- Platzkomplexität?

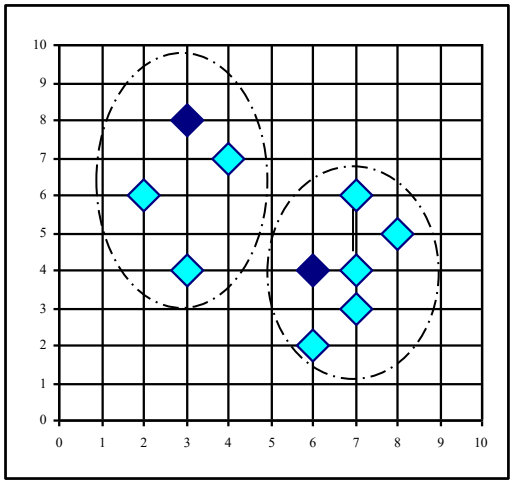
An Efficient k-Means Clustering Algorithm: Analysis and Implementation, Tapas Kanungo, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, Angela Y. Wu. IEEE Trans. On Pattern Analysis and Machine Intelligence, 24(7), **2002**

Weiterer Ansatz: Mehrskalen-k-d-Bäume

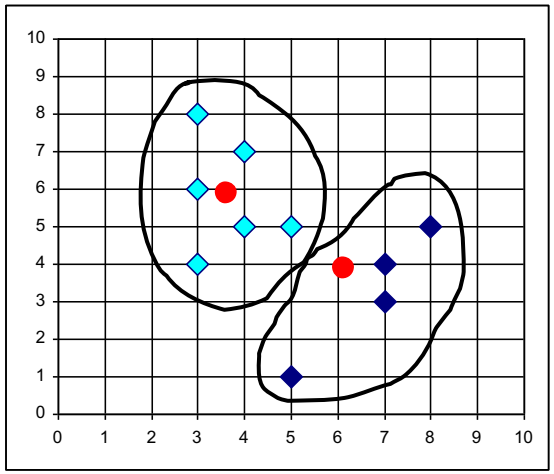
A. Moore, A Very Fast EM-Based Mixture Model Clustering Using Multiresolution kd-Trees, Proc. Conf. Neural Information Processing Systems, **1998**

Partitioning Around Medoids (PAM) Algorithmus

- K-Medoids-Clusterbildung
- Als jeweils (neuer) Zentroid wird ein *Datenpunkt* bestimmt, der se_K minimiert
- Manhattan-Distanz, sonst ähnlich wie K-Means



K-Medoids



K-Means

S. Vinod: Integer programming and the theory of grouping. In: Journal of the American Statistical Association. Band 64, **1969**

Kaufman, L. and Rousseeuw, P.J. , Clustering by means of Medoids, in Statistical Data Analysis Based on the L1–Norm and Related Methods, 405–416, **1987**

PAM-Algorithmus

- Wähle k Objekte als initiale Medoid-Kandidaten $O(1)$
- Solange bis sich nichts mehr ändert #Iterationen * $O((n-k)^2*k)$
  - Weise Objekte dem nächsten Medoiden als Clusterrepräsentanten zu $O((n-k)*k)$
  - Für jedes Paar (Medoid, Nicht-Medoid) $(n-k)*k$ mal
    - Prüfe, ob Tausch se_K kleiner macht $O(n-k)$

$$se_{K_i} = \sum_{j=1}^{m_i} \| t_{ij} - C_i \|_2$$

$$se_K = \sum_{j=1}^{k} se_{K_j}$$

  - Für Paar mit kleinsten Kosten, führe Tausch durch $O(1)$

Was ist das Problem mit PAM?

- PAM robuster als K-Means bzgl. Rauschen und Ausreißern (Medoide weniger beeinflusst durch Ausreißer als Mittel)
- PAM arbeitet effizient auf kleinen Datenmengen aber skaliert nicht besonders gut für große Datenmengen
  - $O(k(n-k)^2) \approx O(k\, n^2)$ da $n >> k$
    für *jede* Iteration (n = #Punkte, k = #Cluster)

➔ Stichproben-basierte Methoden

CLARA (Clustering LARge Applications)

CLARA (Clustering Large Applications) (1986)

- Bestimme mehrere Stichproben aus Datensatz
- Wende PAM auf jede Stichprobe an
- Gebe beste Clusterbildung bzgl. se_K als Ergebnis aus
- Bestimme Cluster in Grundgesamtheit über Voronoi-Diagramm
- Stärken: Größere Datenmengen handhabbar
- Schwächen:
  - Effizienz hängt von Stichprobengröße ab
  - Gute Clusterbildung auf Stichprobe ergibt nicht notwendigerweise gute Clusterbildung auf Grundgesamtheit

Leonard Kaufman, Peter J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, **1990**

Kaufman, Leonard & Rousseeuw, Peter. Clustering Large Data Sets. In Pattern Recognition in Practice II, Elsevier/North Holland 425-437, **1986**

CLARANS ("Randomized" CLARA) *(1994)*

- *CLARANS* (A Clustering Algorithm based on Randomized Search)
- Neue Parameter: maxneighbor und numlocal
- Höchstens maxneighbor viele von zufällig ausgewählten Paaren (Medoid, Nicht-Medoid) werden betrachtet
- Die erste Ersetzung, die überhaupt eine Reduzierung des se_K-Wertes bewirkt, wird auch durchgeführt
- Die Suche nach k „optimalen" Medoiden wird numlocal mal wiederholt (also nicht bis zur Sättigung durchgeführt)
- CLARANS bei angemessener Wahl der Parameter oft nicht viel schlechter, aber viel schneller

R. Ng and J. Han. "Efficient and effective clustering method for spatial data mining". In: Proceedings of the 20th VLDB Conference, pages 144–155, **1994**

Raymond T. Ng and Jiawei Han. CLARANS: A Method for Clustering Objects for Spatial Data Mining. IEEE Trans. on Knowl. and Data Eng. 14, 5, pp. 1003-1016, **2002**

Clusterbildung: Übersicht

- Partitionierungsbasiert
  - K-Means
  - K-Medoids
  - BFR
  - DBSCAN
- Hierarchisch (zusammenballende Clusterbildung)
  - CURE
  - BIRCH
  - Spektrale Clusterbildung

Naive Inkrementelle Nächste-Nachbarn-Clusterbildung

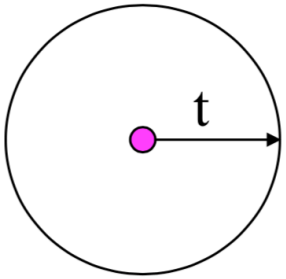
- Punkte werden iterativ und inkrementell in die nächstliegenden Cluster integriert

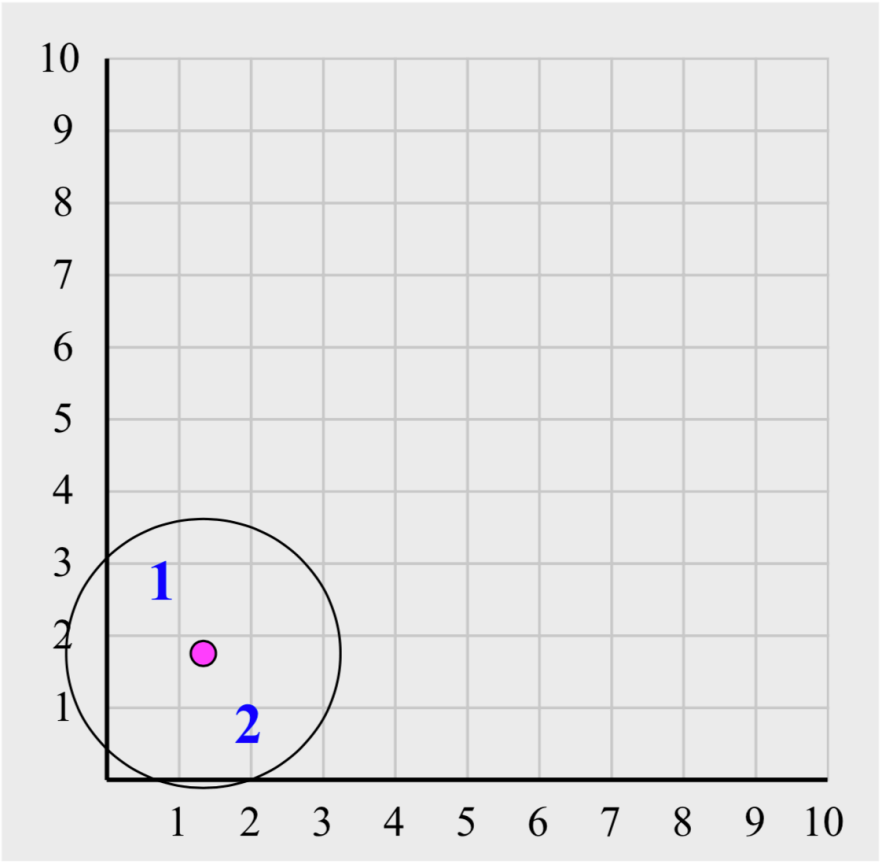
- Schwellwert t zur Bestimmung,
  - ob Punkt integriert wird, oder
  - ob neuer Cluster eröffnet wird

- Nicht verwechseln mit Nächste-Nachbarn-Anfragen

Inkrementelle Nächste-Nachbarn-Clusterbildung

Schwellwert t

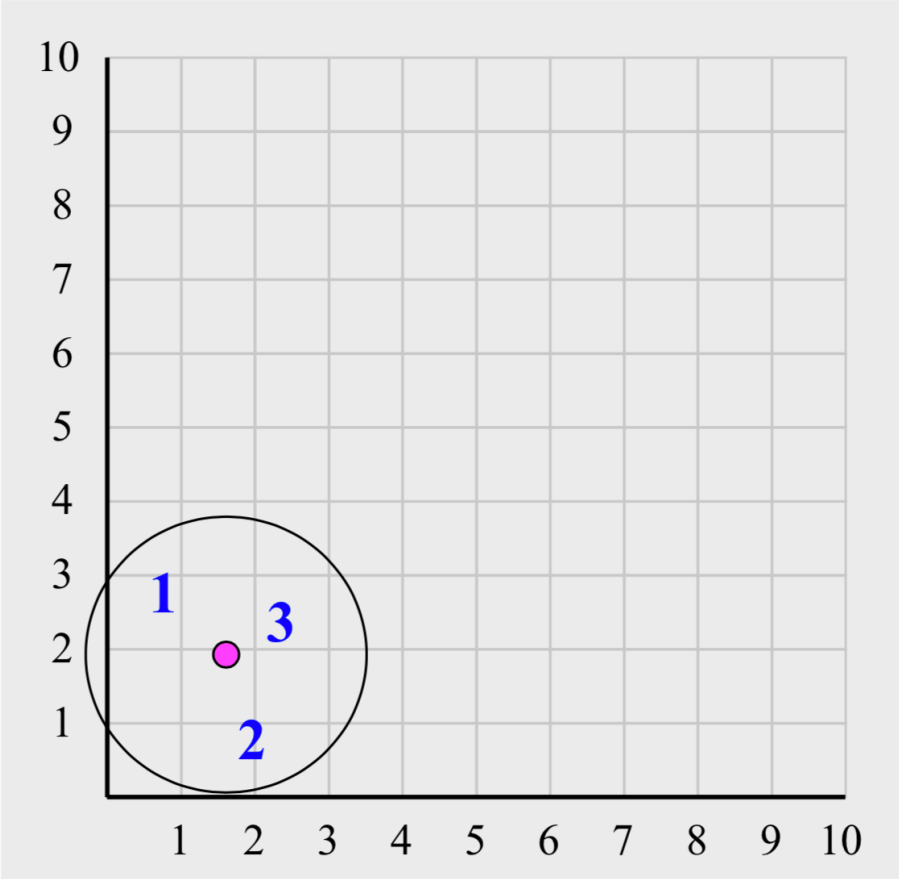




Inkrementelle Nächste-Nachbarn-Clusterbildung

Eintreffen eines neuen Datenpunkts

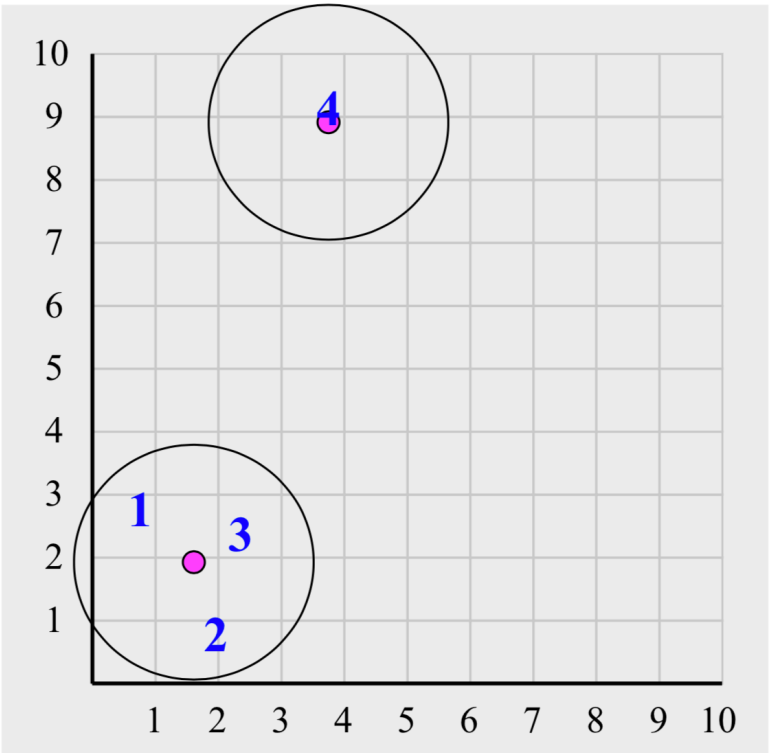
- Hier im Schwellwertbereich von Cluster 1
- Füge zu Cluster 1 hinzu
- Aktualisiere Zentroiden



Inkrementelle Nächste-Nachbarn-Clusterbildung

Eintreffen eines neuen Datenpunkts

- Hier nicht im Schwellwertbereich von Cluster 1
- Erzeuge neues Cluster



Algorithmus ist reihenfolgeabhängig

Schwellwert schwer zu bestimmen

Non-Standard-Datenbanken und Data Mining

Clusterbildung / Häufungsanalyse

Prof. Dr. Ralf Möller

Universität zu Lübeck

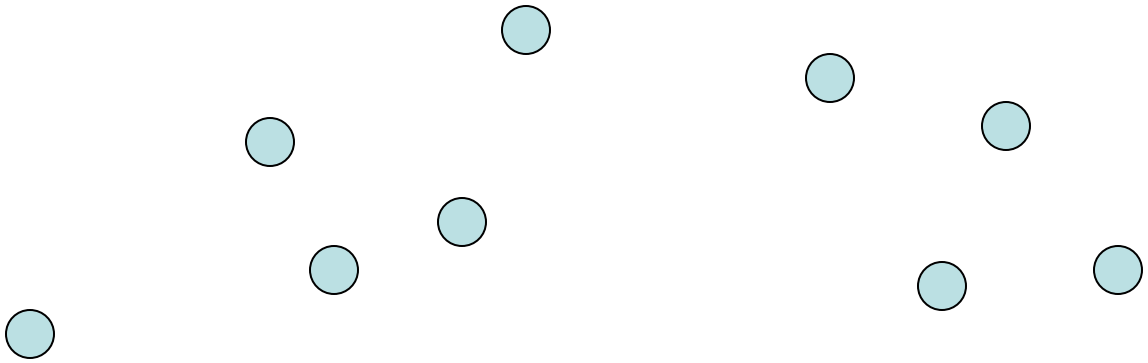
Institut für Informationssysteme

Clusterbildung mit Minimum Spanning Tree (MST)

- Gegeben eine Menge von Punkten in einem r-dimensionalen Raum und eine Zahl k
- Aufgabe: Zerlege die Punktmenge in k Partitionen $S_1,\ldots, S_k$ (Cluster), so dass

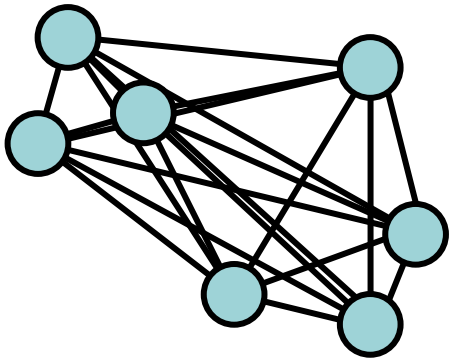
  $$\text{dist}(S_i, S_j) = \min(\{\text{dist}(x, y) \mid x \text{ in } S_i, y \text{ in } S_j\})$$

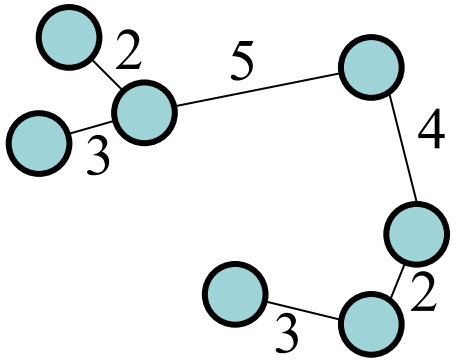
  für alle Paare S_i, S_j $(i < j)$ maximiert wird



Custerbildung mit MST

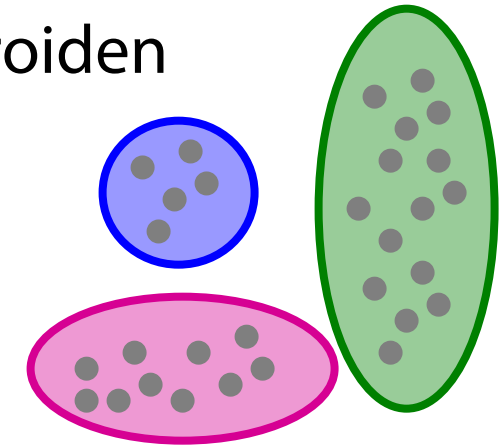
- MST-Berechnung auf vollständigem Graphen mit Gewicht der Kante $\{n_1, n_2\}$ über $dist(n_1, n_2)$ bestimmt (vollst. Graph: #Kanten $m = n^2$)
  - $O(m \log m)$ [Kruskal]
  - $O((m+n)\log n)$ [Jarnik-Prim]
- Entferne k-1 teuerste Kanten





BFR Algorithmus

- BFR [Bradley-Fayyad-Reina] ist eine Variante von *K*-Means für sehr große Datenmengen im Sekundärspeicher
- Annahme**:** Cluster normalverteilt um Zentroiden im Euklidischen Raum
  - Standardabweichungen in verschiedenen Dimensionen können variieren
  - Cluster sind achsenparallele Ellipsen



- Idee: Effizienter Ansatz, Cluster zu codieren
  Ziel: Platzkomplexität O(c) und nicht O(n)
  (c = Anzahl Cluster, n = Anzahl Datenpunkte)

P.S. **B**radley, U.M. **F**ayyad, C. **R**eina, Scaling Clustering Algorithms to Large Databases, Proc. Knowledge Discovery and Data Mining, pp. 9-15, **1998**

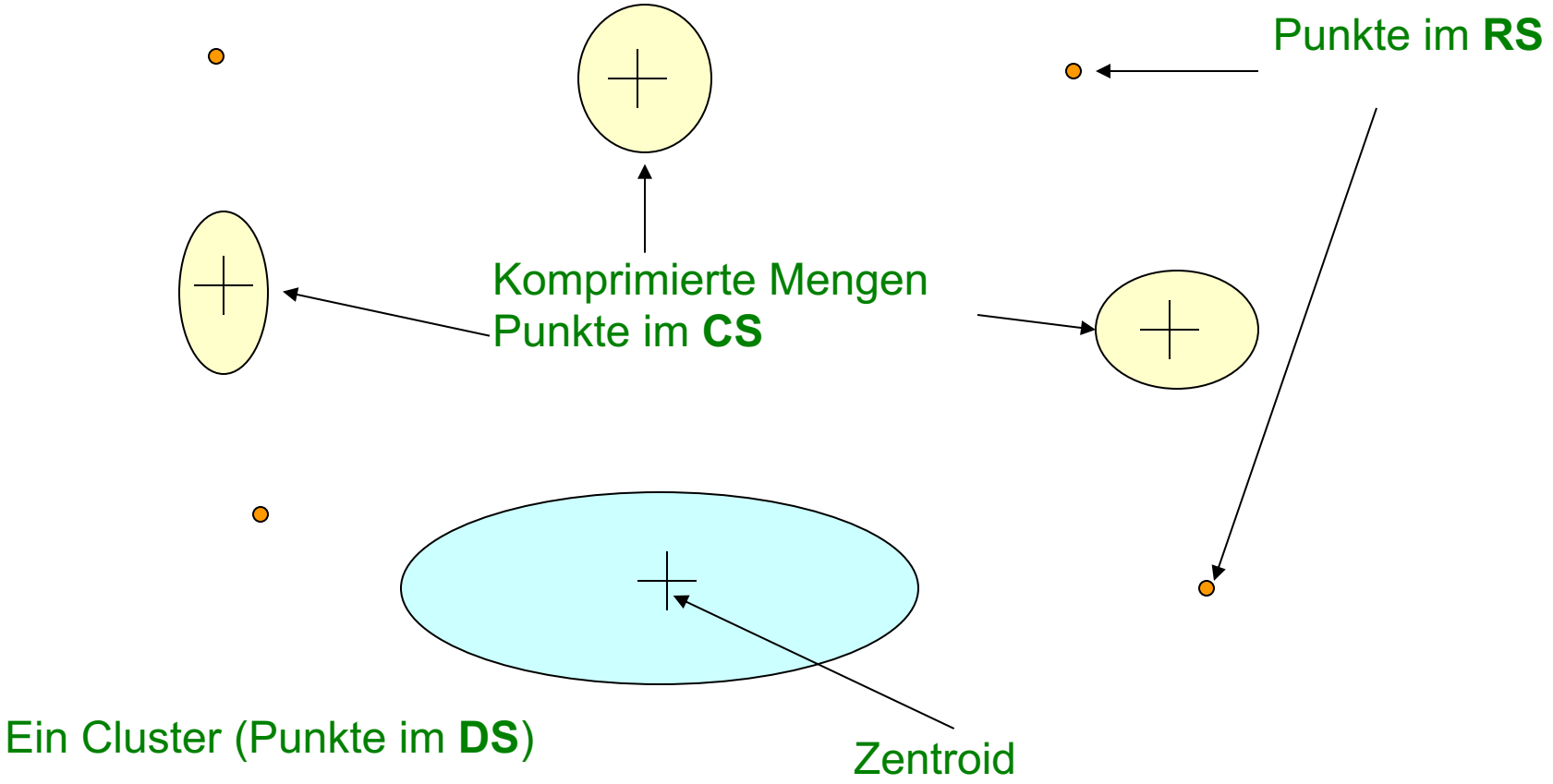
BFR Algorithmus

- Menge von Punkten wird aus dem Sekundärspeicher in Hauptspeicherbereich gelesen bis dieser gefüllt („Punkteladung“)
- Ziel: Punkte (von vorigen Durchgängen) durch einfache Statistiken zusammenfassen
- Wähle initial k Zentroide durch plausible Technik (s.o.)

3 Klassen von Punkten

- **Discard set (DS):**
  - Punkte dicht genug an einem Zentroiden, um in die Zusammenfassung zu kommen
- **Compression set (CS):**
  - Gruppen von Punkten dicht genug zusammen aber nicht dicht genug an einem Zentroiden
  - Punkte werden zusammengefasst, aber nicht einem Cluster zugeordnet
- **Retained set (RS):**
  - Isolierte Punkte, die auf Zuweisung zu einem CS warten

BFR: Bild der “Galaxien”

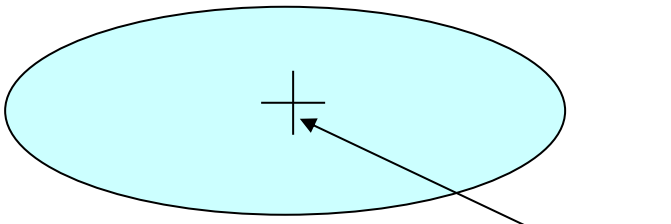


Discard set (DS): Zusammengefasst, in einem Cluster
Compression set (CS): Zusammengefasst, nicht in einem Cluster
Retained set (RS): Isolierte Punkte

Zusammenfassung von Punkten

Für jedes Cluster wird die Menge DS zusammengefasst durch:

- Anzahl der Punkte: *N*
- Vektor *SUM*, dessen i-te Komponente die Summe der Koordinaten der i-ten Dimension ist
- Vektor *SUMSQ*: i-te Komponente = Summe der Quadrate der Koordinaten der i-ten Dimension



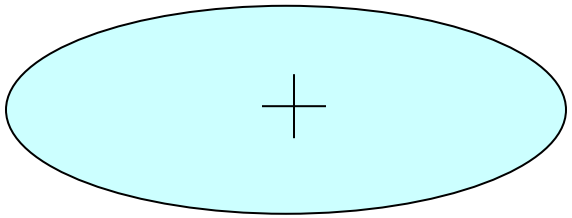
Ein Cluster (Punkte im **DS**)

Zentroid

Zusammenfassung von Punkten: Kommentare

- 2d + 1 Werte repräsentieren Cluster jeder Größe
  - d = Anzahl der Dimensionen
- Mittel in jeder Dimension (Zentroid) kann berechnet werden als SUM_i / N
  - SUM_i = i-te Komponente von SUM
- Varianz von DS eines Clusters in der Dimension i ist:
  $1/N\ (SUMSQ_i - 2\ SUM_i\ (SUM_i / N)\ + (SUM_i / N)^2\ )$
  $= 1/N\ ((N-2)SUMSQ_i / N\ + (SUM_i / N)^2\ )$
  - Standardabweichung ist Wurzel davon
- Nächster Schritt: Tatsächliche Clusterbildung

NB: Wenn wir Achsenparallelität fallen lassen, müssen wir die volle Kovarianzmatrix zur Zusammenfassung bestimmen. Anstelle eines d-dimensionalen Vektors ist **SUMSQ** dann eine ***d x d*** Matrix, was sehr groß wäre!



$$\text{Emp. Var} = 1/N \sum_1^n (x_i - \bar{x})^2$$

Verarbeitung der „Punkteladung“ (1)

1. Finde Punkte, die „genügend dicht“ zu einem Clusterzentroiden sind und füge sie zu diesem Cluster und zu dessen DS hinzu
   - Diese Punkte sind so dicht am Zentroiden, dass sie zusammengefasst und dann entfernt werden können
2. Verwende einen Hauptspeicher-Clusteralgorithmus um die übrigen Punkte und das alte RS zu verarbeiten
   - Cluster kommen nach CS
   - Ausreißerpunke kommen nach RS

Discard set (DS): Zusammengefasst, in einem Cluster
Compression set (CS): Zusammengefasst, nicht in einem Cluster
Retained set (RS): Isolierte Punkte

Verarbeitung der „Punktladung“ (2)

3. DS set: Adjustiere Statistiken der Cluster für die neuen Punkte
   - Aktualisiere Ns, SUMs, SUMSQs
4. Versuche CS-Elemente zusammenzufassen
5. Falls letzte Runde, verschmelze all komprimierten Mengen in CS und alle RS Punkte in das jeweils nächste Cluster aus DS set

Discard set (DS): Zusammengefasst, in einem Cluster
Compression set (CS): Zusammengefasst, nicht in einem Cluster
Retained set (RS): Isolierte Punkte

Einige weitere Details…

1. Wie entscheiden wir, wann ein Punkt „dicht genug“ an einem Cluster ist, so dass wir ihn dort hinzufügen und dann entfernen?

2. Wie entscheiden wir, ob Elemente von CS verschmolzen werden sollen?

Ad 1: Wie dicht ist dicht genug?

- Hohe Wahrscheinlichkeit, dass Punkt zum aktuell dichtesten Zentroiden gehört
  - Akzeptiere Punkt in Cluster, wenn Mahalanobis-Distanz kleiner als Schwellwert, z.B. zwei Standardabweichungen

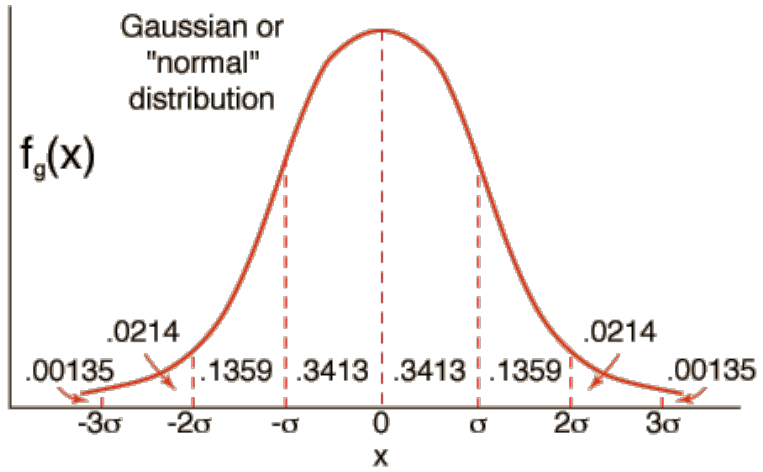
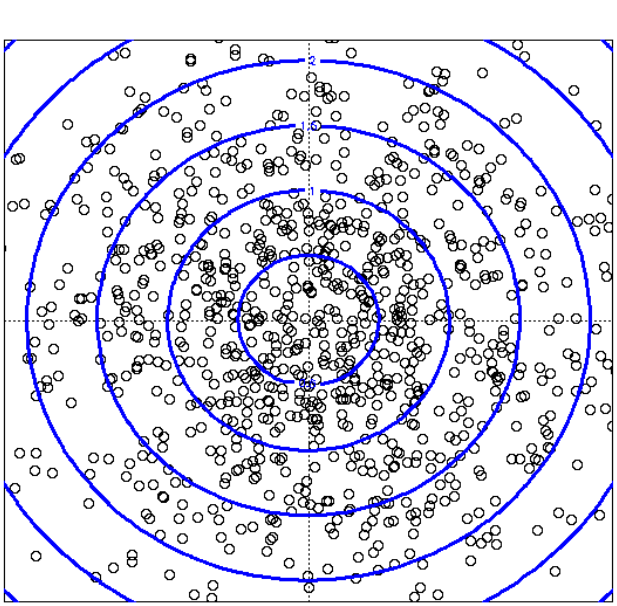


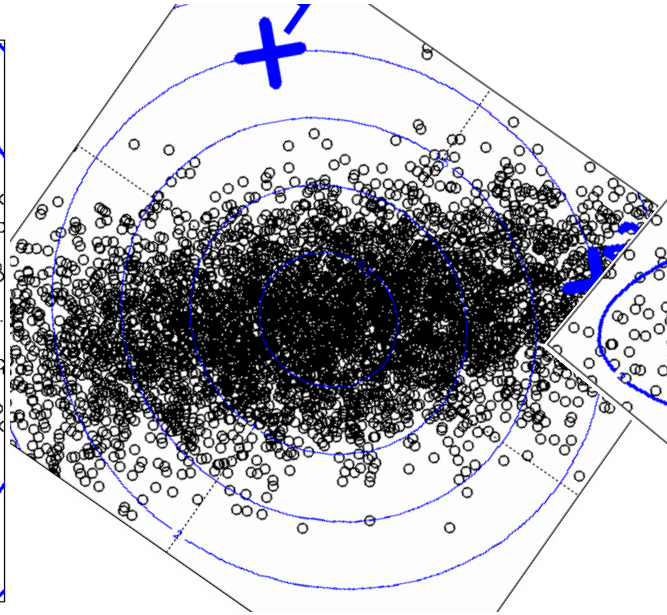
Illustration: Gleiche M.D. Linien

- **Euklidische Distanz vs. Mahalanobis-Distanz**

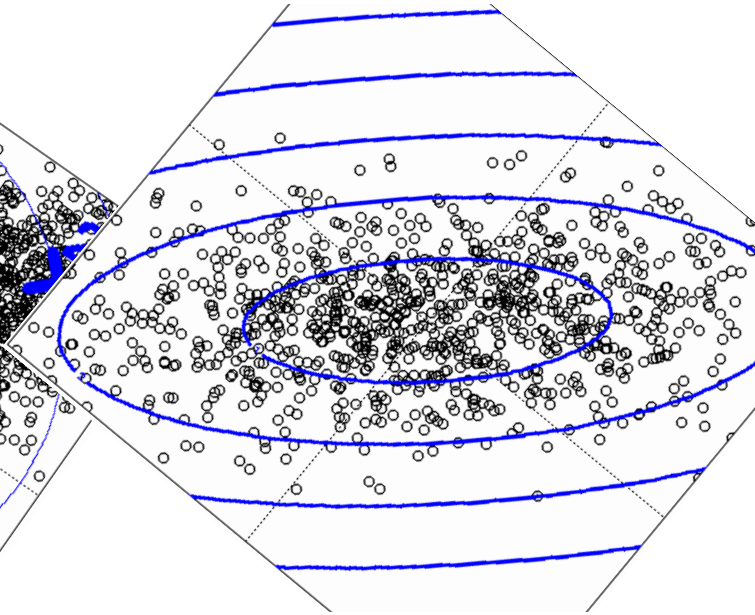
Konturen von Punkten mit gleichem Abstand zum Ursprung



Gleichverteilte Punkte
Euklidische Distanz



Normalverteilte Punkte
Euklidische Distanz



Normalverteilte Punkte
Mahalanobis-Distanz

Mahalanobis-Distanz

- **Normalisierter Euklidischer Abstand vom Zentroiden**

- Für Punkt $(x_1, \ldots, x_d)$ und Zentroid $(c_1, \ldots, c_d)$
  1. Normalisiere in jeder Dimension: $y_i := (x_i - c_i) / \sigma_i$
  2. Nehme Summe der Quadrate der y_i
  3. Bestimme Wurzel

$$d(x, c) = \sqrt{\sum_{i=1}^{d} \left(\frac{x_i - c_i}{\sigma_i}\right)^2}$$

σ_i … Standardabweichung der Punkte im Cluster in der i-ten Dimension (siehe oben)

Ad2: Sollen CS-Cluster verschmolzen werden?

- Bestimmung der Standardabweichung der kombinierten Untercluster aus ***N***, ***SUM***, und ***SUMSQ*** ermöglicht effiziente Berechnung
- Kombinierung, wenn Mahalanobis-Abstand des Zentroiden des einen Clusters unter Schwellwert bzgl. Standardabweichung des anderen Clusters

Anforderungen:

- Cluster müssen um Zentroiden normalverteilt sein
- Dimensionen müssen unabhängig sein

Keine Garantie, dass Reihenfolgeabhängigkeiten nicht auftreten

P.S. **B**radley, U.M. **F**ayyad, C. **R**eina, Scaling Clustering Algorithms to Large Databases, Proc. Knowledge Discovery and Data Mining, pp. 9-15, **1998**

Clusterbildung: Übersicht

- Partitionierungsbasiert
  - K-Means
  - K-Medoids
  - BFR
  - DBSCAN
- Hierarchisch (zusammenballende Clusterbildung)
  - CURE
  - BIRCH
  - Spektrale Clusterbildung

Dichtebasierende partitionierende Clusterbildung

- DBSCAN-Verfahren (Density Based Spatial Clustering of Applications with Noise)
  - Dichtebasierte räumliche Clusteranalyse unter Berücksichtigung von Rauschen
- 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

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), S. 226–231, **1996**

DBSCAN – Definitionen

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

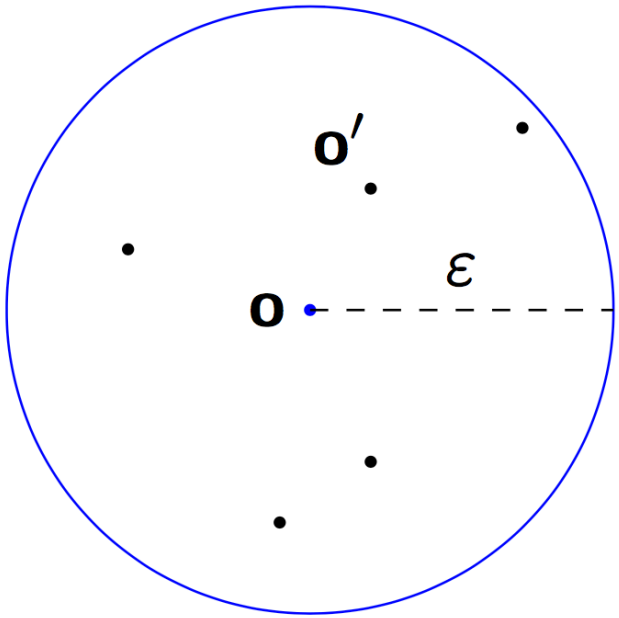
$$N_\varepsilon(\mathbf{o}) := \{\mathbf{o}' \in O : d(\mathbf{o}, \mathbf{o}') \leq \varepsilon\}$$

- 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

- im Folgenden sei core(O) die Menge aller Kernobjekte in O
- in den folgenden Beispielen: $m = 4$

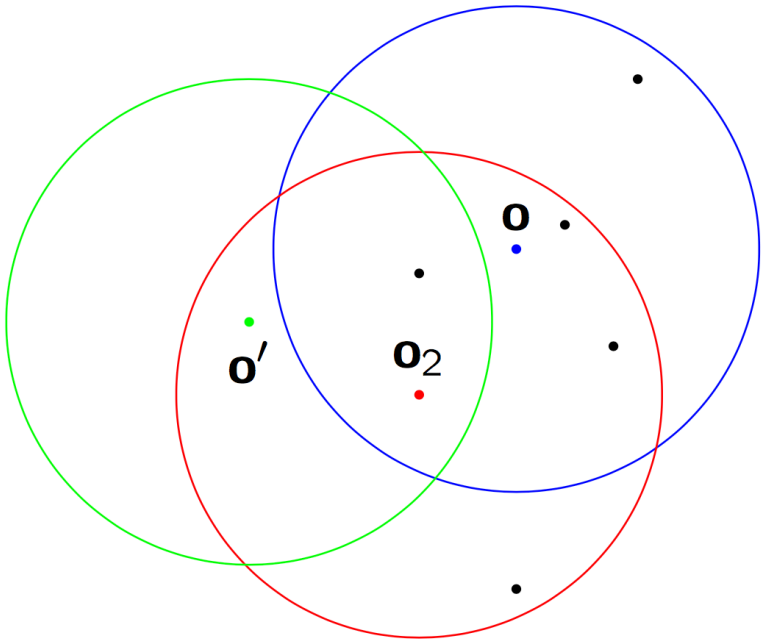
DBSCAN – Definitionen

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



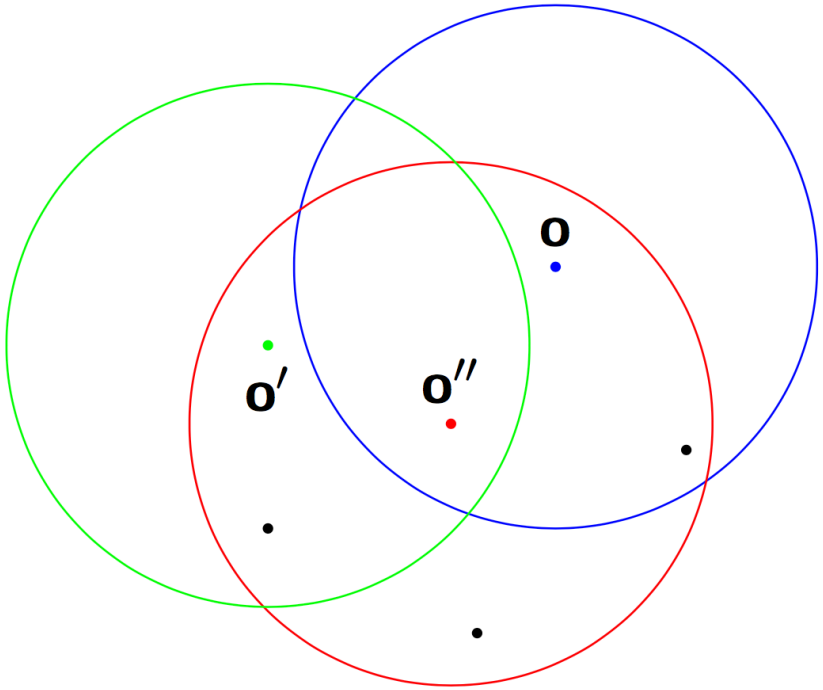
DBSCAN – Definitionen

$\mathbf{o}'$ 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 $(\varepsilon\ , m\ )$ 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.

Nicht alle Punkte im Cluster sind Kernobjekte

DBSCAN – Lemma 1

Lemma 1

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

Dichte-Erreichbarkeit ist reflexiv, also ist Kernobjekt ***o*** auch im Cluster

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.

Damit haben wir weniger Probleme mit Reihenfolgeabhängigkeiten

DBSCAN

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

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 // $\mathbf{o}$ ist ein Randobjekt
  $c(\mathbf{o}) := -2$
  return false
else // $\mathbf{o}$ ist ein Kernobiekt
  // 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
    $\mathbf{o}' :=$ *getElement(S)*
    $R := neighborhood(O, \mathbf{o}', \varepsilon)$
    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

Laufzeitkomplexität von DBSCAN hängt von der Komplexität von neighborhood ab

Nutze multidimensionalen Index

DBSCAN mit k-d-Bäumen

Sei n die Anzahl der Punkte

- $O(n \cdot \log n)$ Zeitkomplexität durch effektiven k-d-Baum mit ε-Regionenanfrage (neighborhood) statt $O(n^2)$ ohne Indexstrukturen
- $O(n)$ Platzkomplexität

Vorteile

- Effizient für niedrigdimensionale Daten
- Cluster willkürlicher Gestalt erkennbar
- Robust gegen Ausreißer (Rauschen)

DBSCAN: Probleme

- DBSCAN ist sensitiv bezüglicher der Parameter
  - m (minimale Anzahl Nachbarn für Kernobjekte) und
  - ε
- Nicht einfach geeignet für nebenläufige Verarbeitung
- m ist fix: DBSCAN erkennt Cluster bei variierender Dichte nicht besonders gut

Clusterbildung: Übersicht

- Partitionierungsbasiert
  - K-Means
  - K-Medoids
  - BFR
  - DBSCAN
- Hierarchisch (zusammenballende Clusterbildung)
  - CURE
  - BIRCH
  - Spektrale Clusterbildung

Zusammenballende hierarchische Clusterbildung

- Angefangen von Clustern mit einem Punkt, verschmelze zwei oder mehr Cluster zu einem größeren Elterncluster bis ein Abbruchkriterium erfüllt ist
- Verschmelzung von „dichtesten“ Clustern
- Gesucht: Definition für Distanz zwischen Clustern

Distanzfunktionen

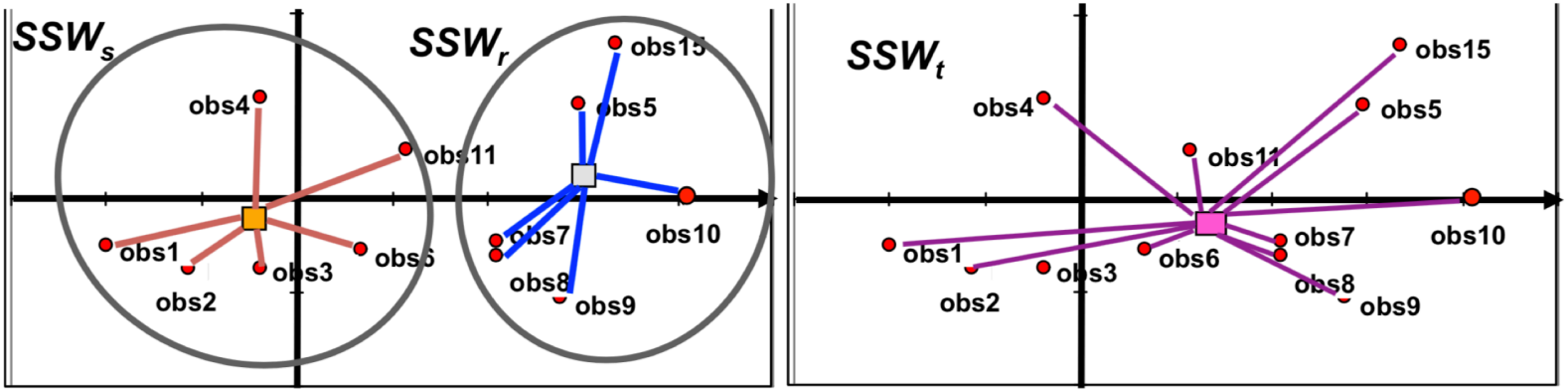
- MIN (Single Link, Distanz zwischen dichtesten Objekten)
- MAX (Complete Link, maximale Distanz zw. Objekten in C_i und Obj. in C_j)
- Gruppenmittel (Group Average, GA, mittlere Distanz zwischen jedem Objekt in C_i und jedem Objekt in C_j)
- Zentroidendistanz zwischen Cluster C_i und C_j ist die Distanz zwischen dem Zentroiden r_i von C_i und dem Zentroiden r_j von C_j
- Ward-Distanz zwischen Cluster C_i und C_j ist die Differenz zwischen der In-Cluster Summe der Quadrate nach dem Verschmelzen von C_i und C_j zu C_{ij} und der Summe der in-Cluster Summen der Quadrate je Cluster separat

$$D_w(C_i, C_j) = \sum_{x \in C_{ij}} (x - r_{ij})^2 - \left(\sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 \right)$$

- r_i: Zentroid von C_i
- r_j: Zentroid von C_j
- r_{ij} Zentroid von C_{ij}

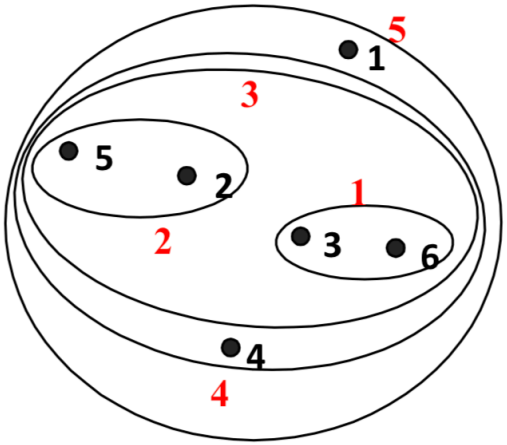
Ward-Distanz

- Vereinige Cluster mit kleinster Ward Distanz



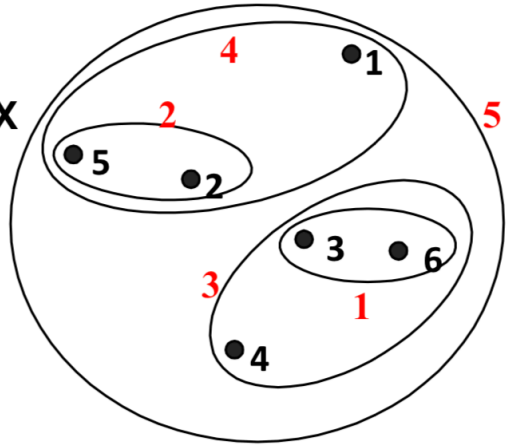
- Ähnlich zu Gruppenmittel und Zentroidendistanz
  - Weniger anfällig bzgl. Rauschen und Ausreißer
  - Generiert vorzugsweise sphärische Cluster

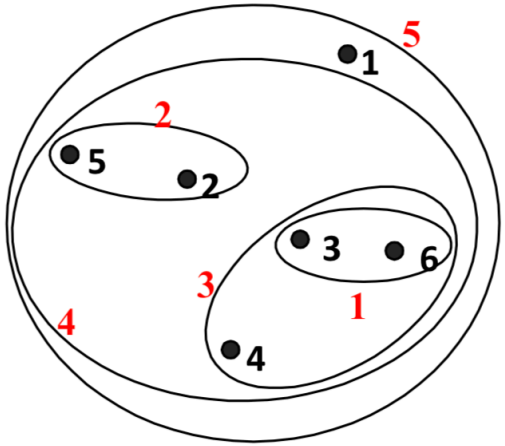
Hierarchische Clusterbildung: Vergleich



MIN

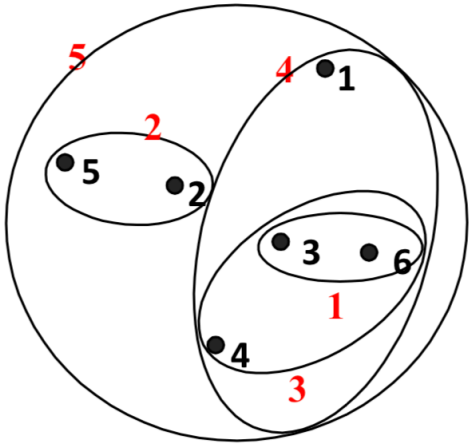
MAX





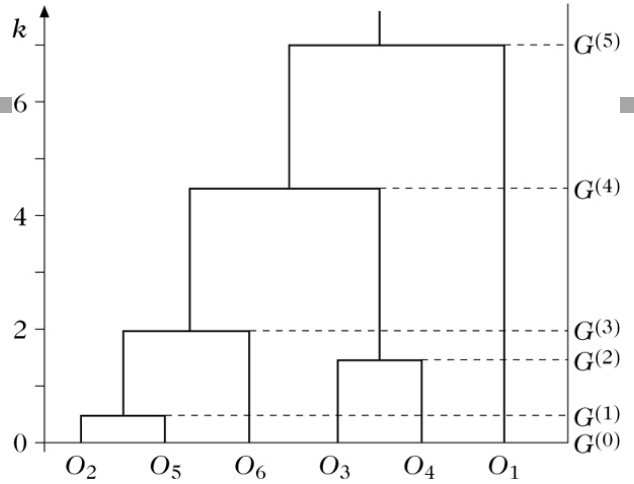
Group Average

Ward's Method



Zeit- und Platzbedarf

Dendrogramm



- Für einen Datensatz mit n Punkten
- Platz: $O(n^2)$
  - Speicherung der Distanzmatrix
- Zeit: $O(n^3)$
  - Es gibt n Schritte, und in jedem Schritt wird die Distanzmatrix der maximalen Größe n^2 aktualisiert
  - Reduktion auf $n^2 \log n$ möglich durch Indexstrukturen

Hat jemand eine Idee?
kd-Bäume?

Hierarchische Clusterbildung: Bewertung

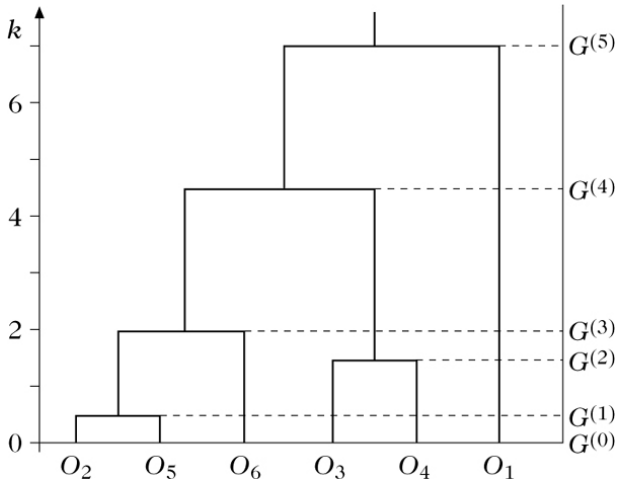
- 😀 Anzahl der Cluster muss nicht vorgegeben werden
- 😟 Skalierbarkeit problematisch
- 😟 Lokale Optima

- 😟 MIN: anfällig für Rauschen oder Ausreißer
- 😟 MAX/GA: Probleme bei nicht-sphärischen Clustern

Clusterbildung: Übersicht

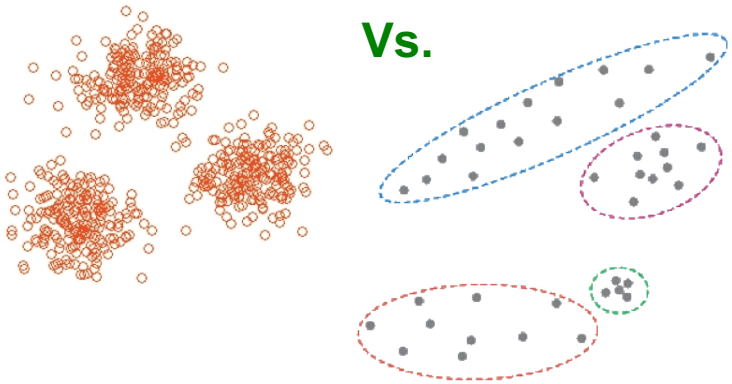
- Partitionierungsbasiert
  - K-Means
  - K-Medoids
  - BFR
  - DBSCAN: O(n log n)
- Hierarchisch
  - Zusammenballende Clusterbildung
    - Von Distanzmaß abhängig
    - Welche Ebene betrachten?
    - $O(n^3)$ bzw. $O(n^2 \log n)$
  - CURE
  - BIRCH
  - Spektrale Clusterbildung

Dendrogramm

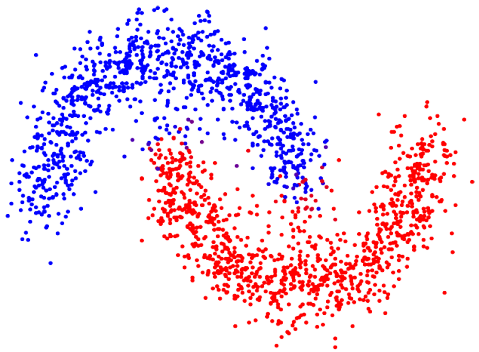


Das CURE-Verfahren

- Vgl. BFR (aus 1998):
  - Annahme: Cluster normalverteilt in jeder Dimension
  - Achsenparallele Ellipsen, gedrehte Ellipsen nicht vorgesehen
  - Blockweise Verarbeitung von Daten
  - Speichereffiziente Beschreibung der Cluster

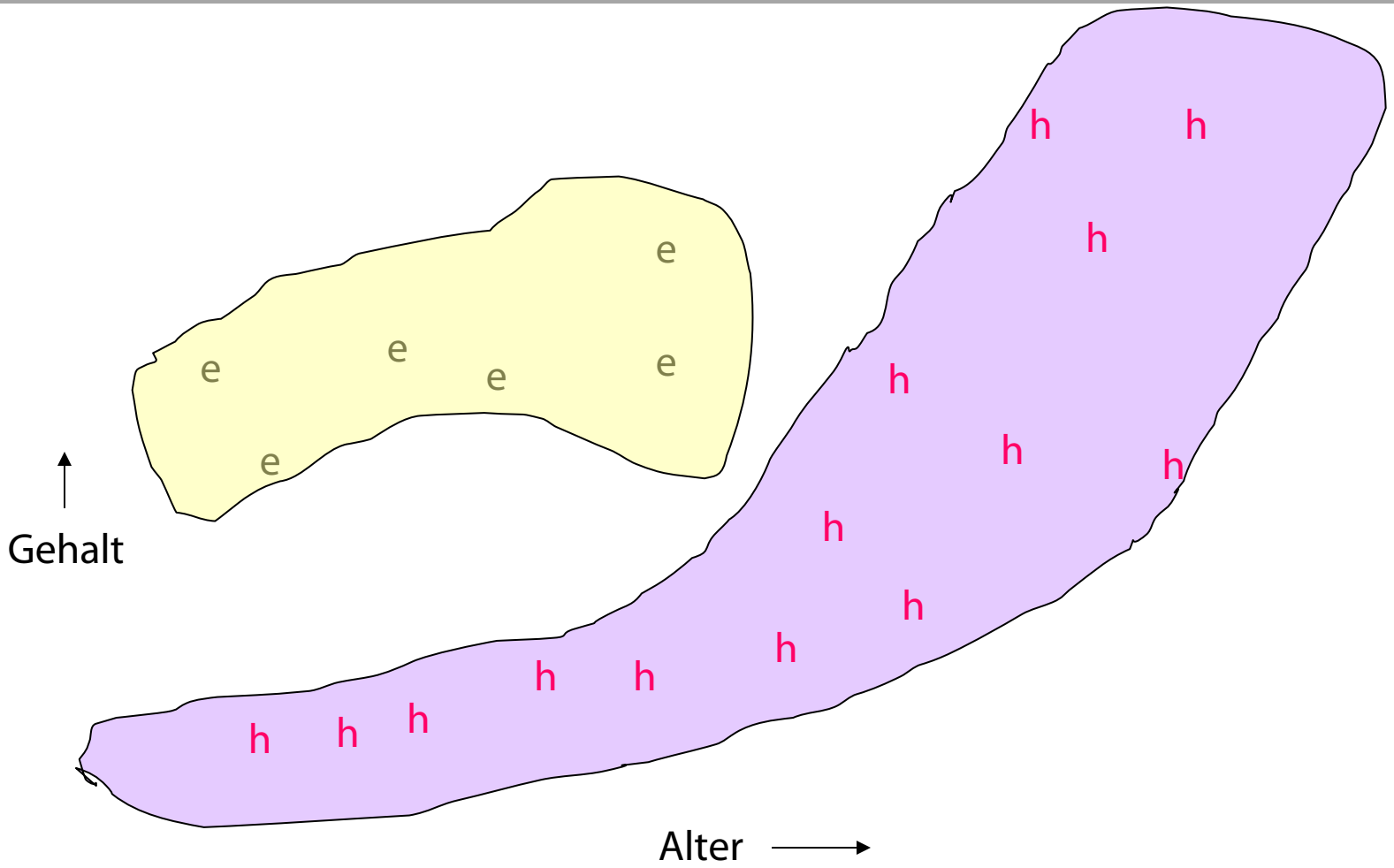


- CURE (Clustering Using Representatives, auch aus 1998):
  - Verwendung einer Stichprobe (zeiteffizient) und einer Menge von repräsentativen Punkten, um Cluster zu repräsentierten (speichereffizient)



Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. 1998. CURE: An efficient clustering algorithm for large databases. SIGMOD Rec. 27, 2 73-84, **1998**

Beispiel

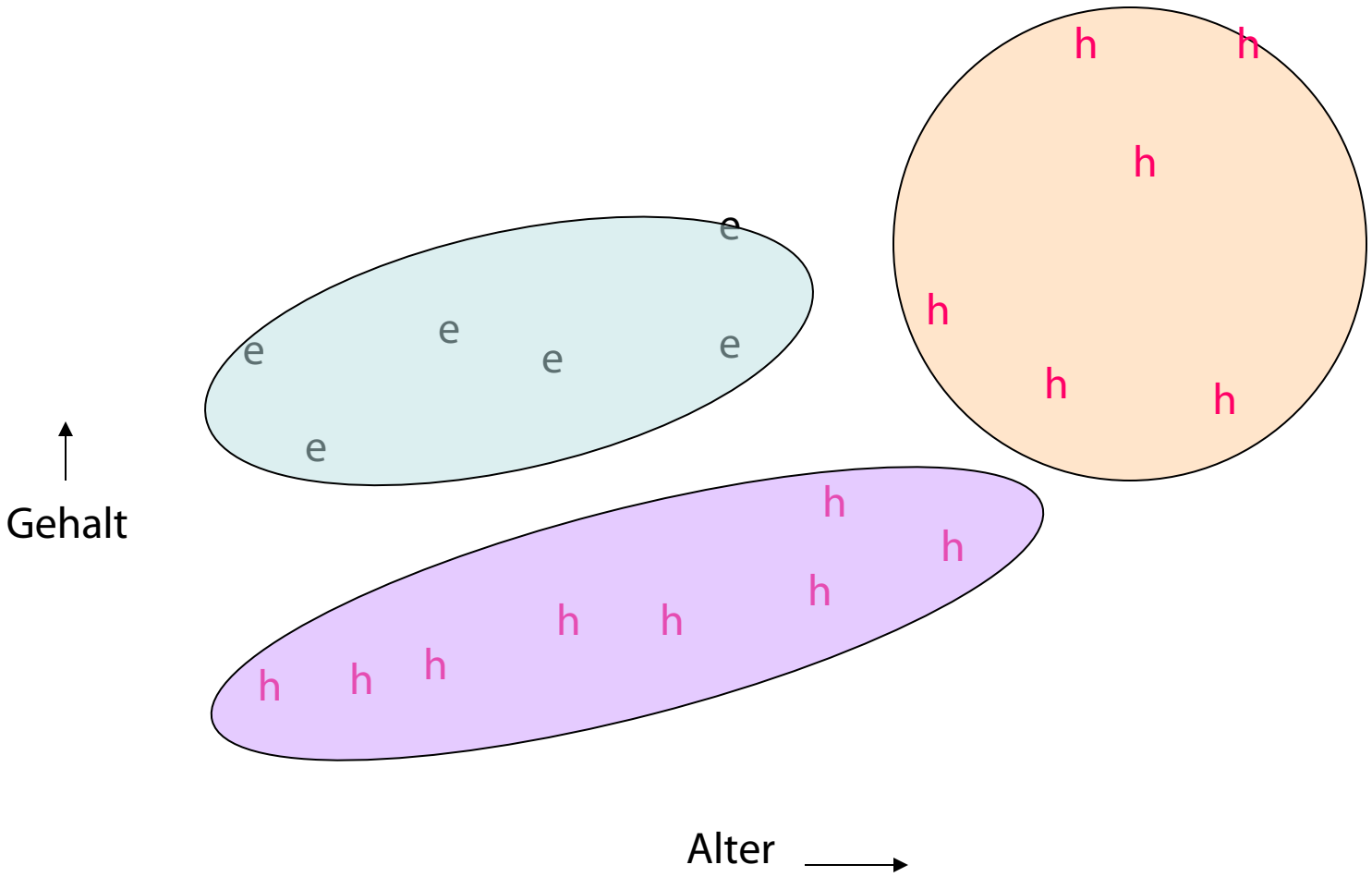


CURE-Idee: 2 Durchgänge

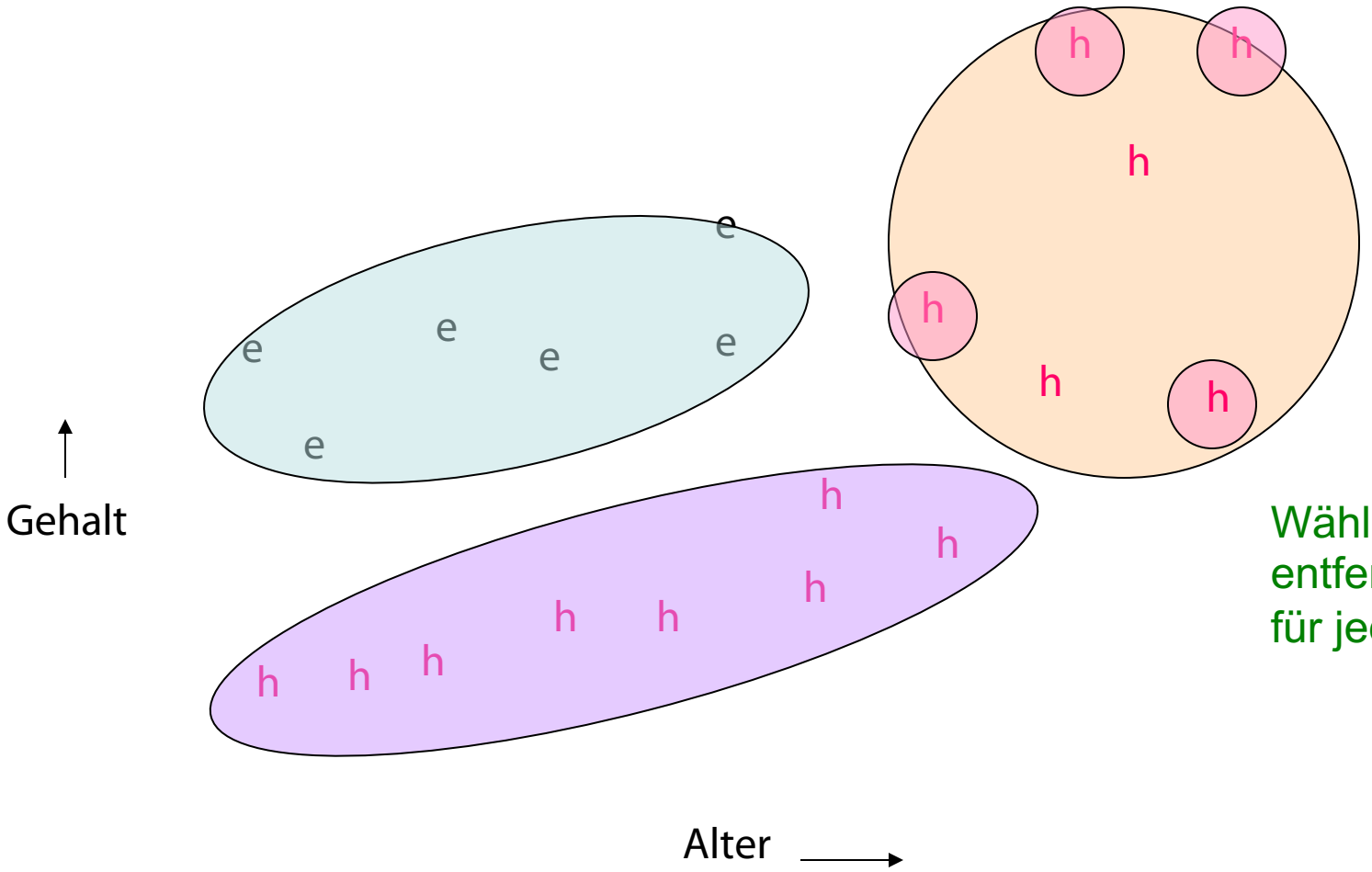
Durchgang 1

1. Wähle Punkt-Stichprobe, die in Hauptspeicher passt
2. Fange an mit „kleinen“ **initialen Clustern**
   - Clusterbildung hierarchisch
3. Wähle **repräsentative Punkte**
   - Für jedes Cluster wähle c Punkte (Repräsentanten) so weit auseinanderliegend wie möglich
   - Für die Stichprobe wähle repräsentative Punkte durch Verschiebung um α (bspw. 20%) zum Zentroiden („Eliminiere Ausreißer“)

Initiale Cluster

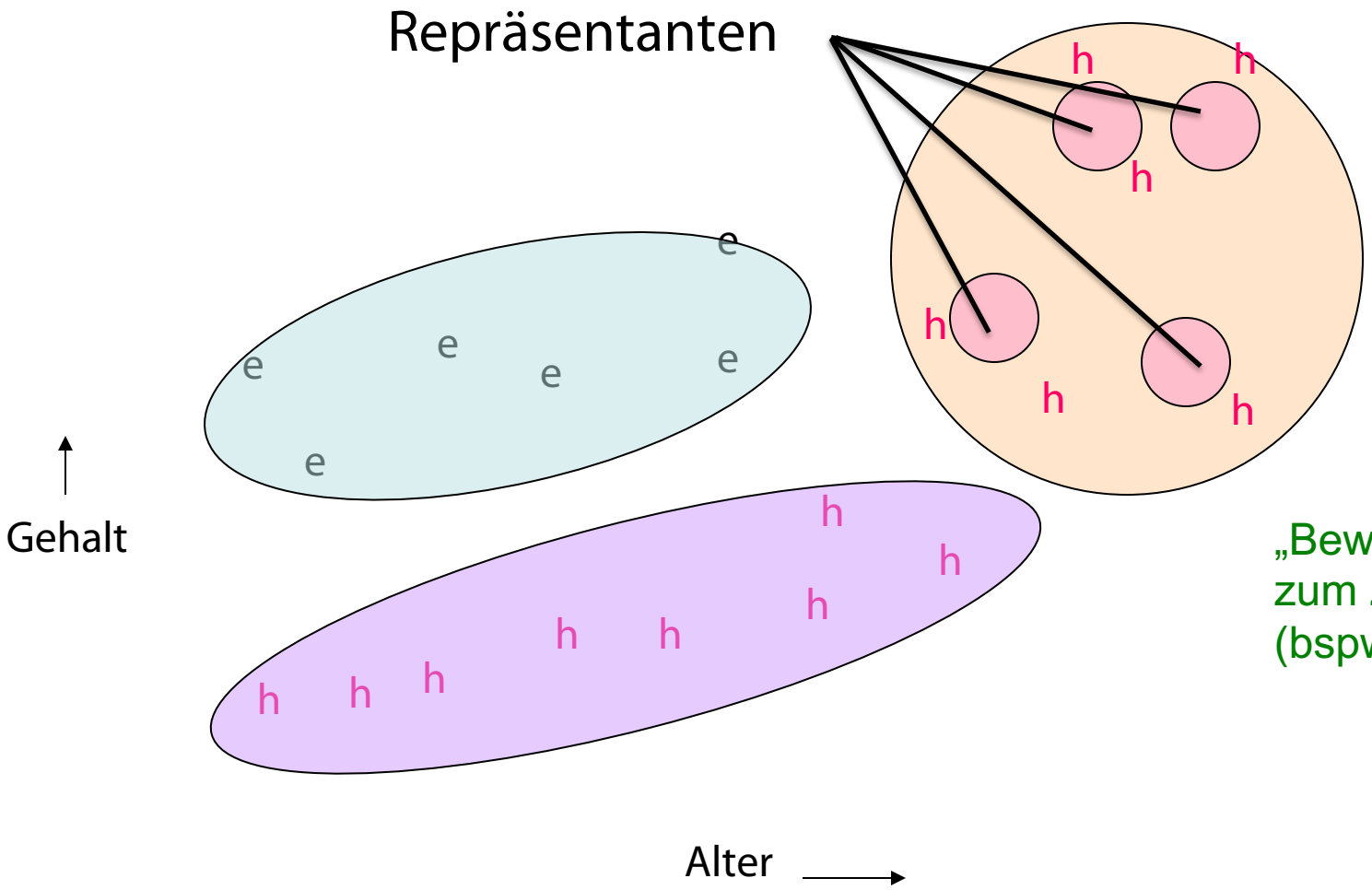


Wähle verteilte Punkte



Wähle bspw. 4 entfernte Punkte für jedes Cluster

Verschiebungsfaktor α

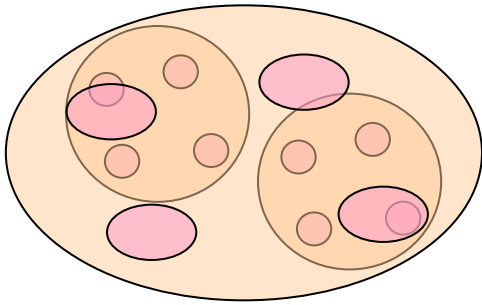


„Bewege“ Punkte zum Zentroiden (bspw. $\alpha = 20\%$)

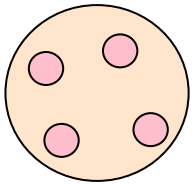
CURE-Idee

Durchgang 2:

- Verschmelze zwei Cluster falls die Repräsentanten „dicht“ (MIN) zusammen liegen
- Wiederhole eventuell bis keine zwei Cluster mit „nahen“ Repräsentanten vorhanden



p



Clusterzuordnung für Daten im Sekundärspeicher

- Betrachte nun alle Punkte p aus dem sekundären Speicher
- Ordne p dem „dichtesten Cluster“ zu:
  Finde dichtesten Repräsentantenpunkt und ordne p in das Cluster des gefundenen Repräsentanten ein

CURE-Algorithmus

- Datensatz S mit n d-dimensionalen Punkten p
- k die gewünschte Anzahl von Clustern
- Für jeden Punkt p zu speichern
  - cluster(p)
- Für jedes Cluster u zu speichern:
  - points(u) // Punkte, die zum Cluster gehören
  - mean(u) // Zentroid
  - rep(u) // Repräsentanten
  - closest(u) // dichtester Cluster zu u
- $dist(u, v) = \min_{p \in rep(u), q \in rep(v)} dist(p, q)$
- $dist(p, q) = L_p$-Distanzmaß oder Ähnlichkeitsmaß

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. 1998. CURE: An efficient clustering algorithm for large databases. SIGMOD Rec. 27, 2 73-84, **1998**

CURE-Algorithmus

```
algorithm cluster (S, k)
   // S sample dataset of points, k number of clusters to be derived
 let T = build_kd_tree(S)              // stores representative points of each cluster
     pq = build_priority_queue('min', map(lambda(p)
                                            closest(p) := NN_kd_tree(p, T)
                                            (p, dist(p, closest(p))),
                                          S) // increasing order of distance between p and closest(p)
     u, v, w, c
   while size(pq) > k and not empty?(pq) do
      u := delete_next(pq); v := closest(u); delete(v, pq); w := merge(u, v)
      delete(rep(u), T); delete(rep(v), T); insert(rep(w), T)
      closest(w) := next(pq)           // just to have some arbitrary initial value
      for x in pq do
         if dist(w, x) < dist(w, closest(w)) then
            closest(w) := x
         if closest(x) ∈ {u, v} then
            if dist(x, closest(x)) < dist(x, w) then
               c := closest(x); closest(x) := closest_cluster(T, x, dist(x, w))
            else
               c := closest(x); closest(x) := w
            decrease_key(x, pq, dist(x, c) – dist(x, closest(x)))
         else
            if dist(x, closest(x)) > dist(x, w) then
               c := closest(x); closest(x) := w
               decrease_key(x, pq, dist(x, c) – dist(x, closest(x)))
         insert((w, dist(w, closest(w))), pq)
   return elements(pq)
```

merge(u, v)

```
function merge (u, v)     // merge two clusters u and v
  let w = u ∪ v
      tmp = ∅
   rep(w) := ∅
   mean(w) := (|u|·mean(u) + |v|·mean(v))/(|u|+|v|)
   for i from 1 to c do          // c = #representatives, a global constant
      max_dist := 0
      for p in points(w) do
         if i = 1 then
            min_dist := dist(p, mean(w))
         else
            min_dist := min({dist(p, q): q ∈ tmp})
         if min_dist ≥ max_dist then
            max_dist := min_dist
            max_point := p
      tmp := tmp ∪ {max_point}
   for p in tmp do
      rep(w) := rep(w) ∪ { p + α·(mean(w)-p) }      // Wenn α = 0, wird CURE zu MST
   return w
```

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. 1998. CURE: An efficient clustering algorithm for large databases. SIGMOD Rec. 27, 2 73-84, **1998**

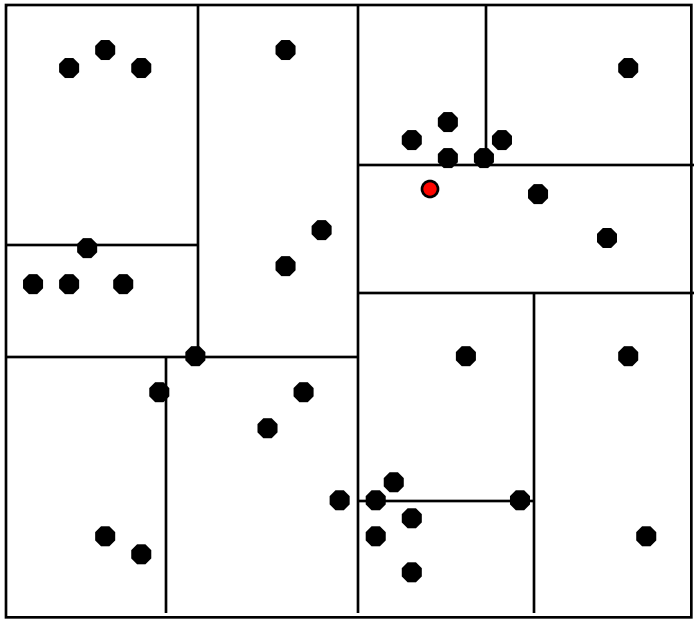
closest_cluster(T, x, max_dist)

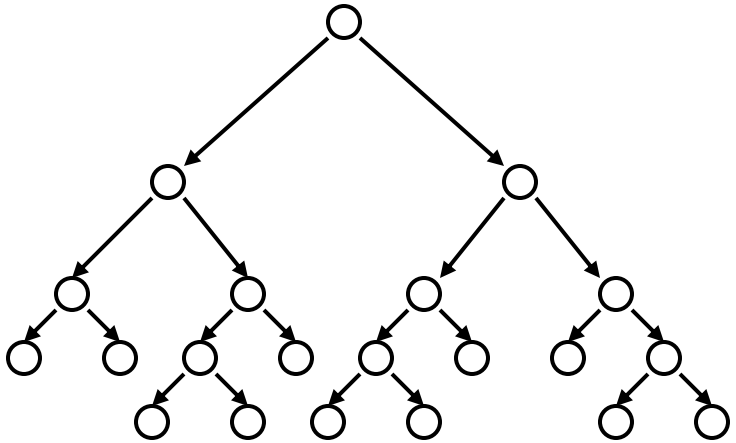
```
function closest_cluster (T, x, max_dist)
 // Verwende T, um zu x nächsten Cluster zu finden
 let n
     q = null
   for p in rep(x) do
      n := nn_kd_tree(p, T, max_dist, lambda(p) p ∉ rep(x))
      if q = null or min_{p∈rep(x)}dist(p, n) < min_{p∈rep(x)}dist(p, q) then
         q := n
   return cluster(q)
```

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. 1998. CURE: An efficient clustering algorithm for large databases. SIGMOD Rec. 27, 2 73-84, **1998**

The procedure closest_cluster (in Step 15) uses the tree T to determine the closest cluster to cluster x. For every point p in x.rep, T is used to determine the nearest neighbor to p that is not in x.rep. From among the nearest neighbors, the one that is closest to one of x's representative points is determined and the cluster containing it is returned as the closest cluster to x. Since we are not interested in clusters whose distance from x is more than $dist(x, w)$, we pass this as a parameter to closest_cluster which uses it to make the search for nearest neighbors more efficient.

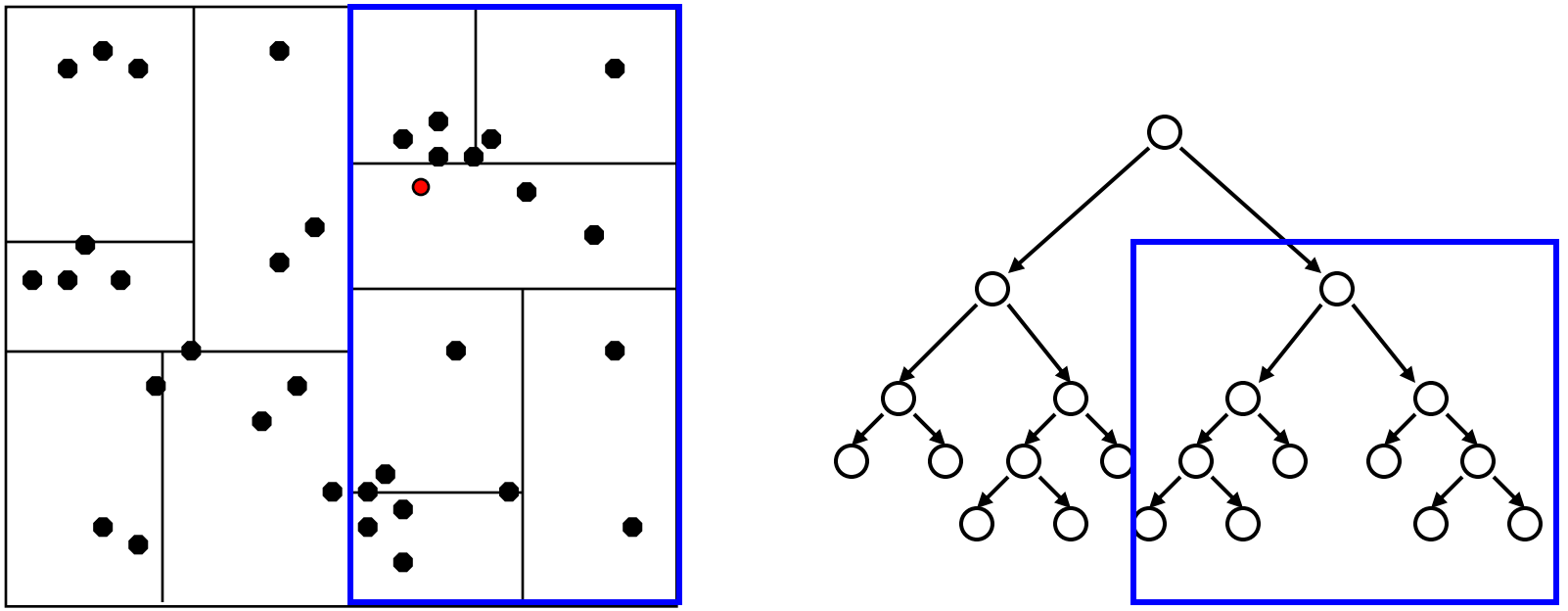
Nächster Nachbar mit k-d-Bäumen





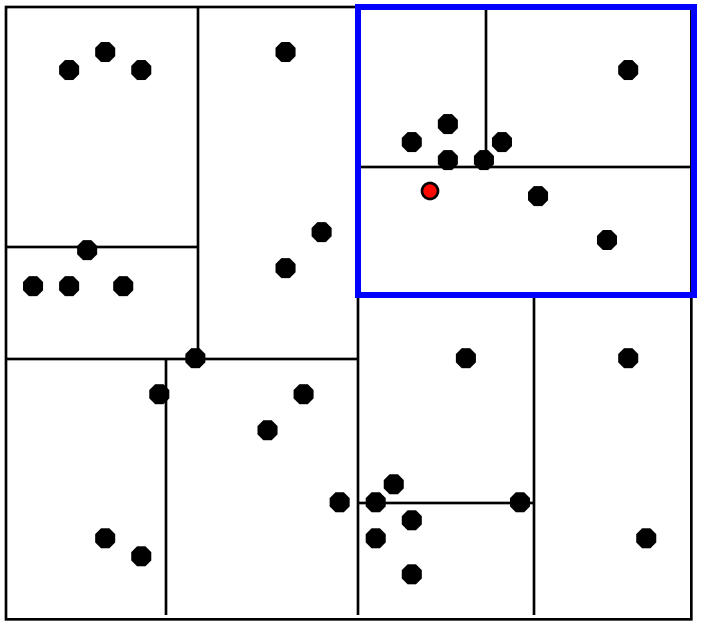
Traversiere Baum, um nächsten Nachbarn
von Anfragepunkt zu finden (Distanzgrenze)

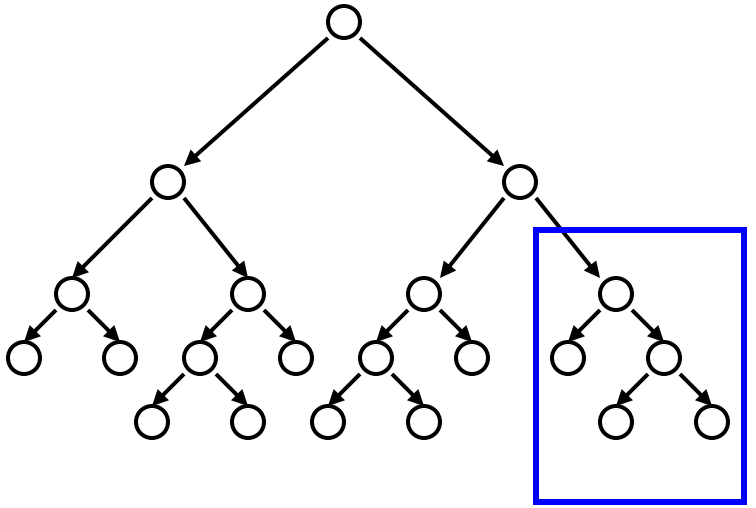
Nächster Nachbar mit k-d-Bäumen



Exploriere Zweig, der dem Anfragepunkt am nächsten kommt

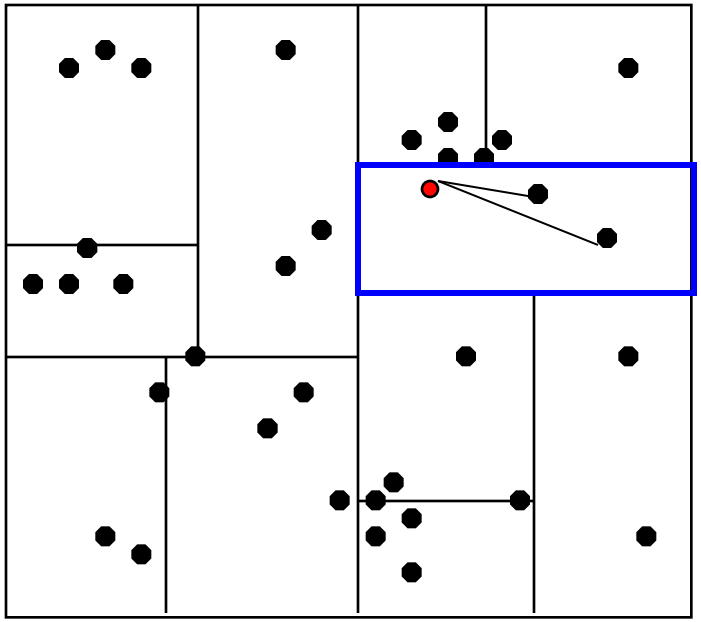
Nächster Nachbar mit k-d-Bäumen

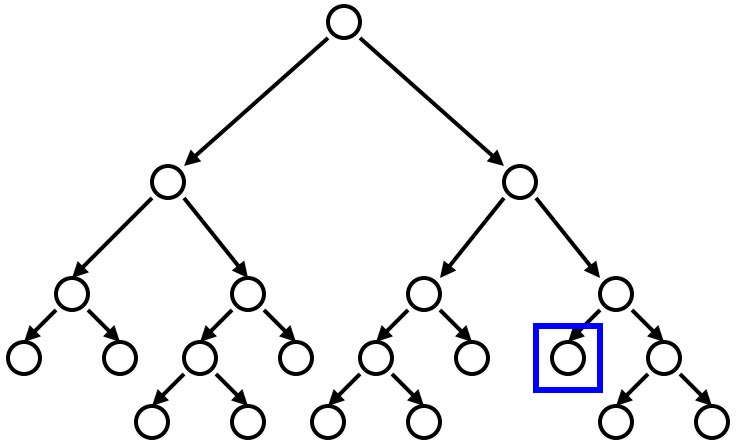




Exploriere Zweig, der dem Anfragepunkt am nächsten kommt

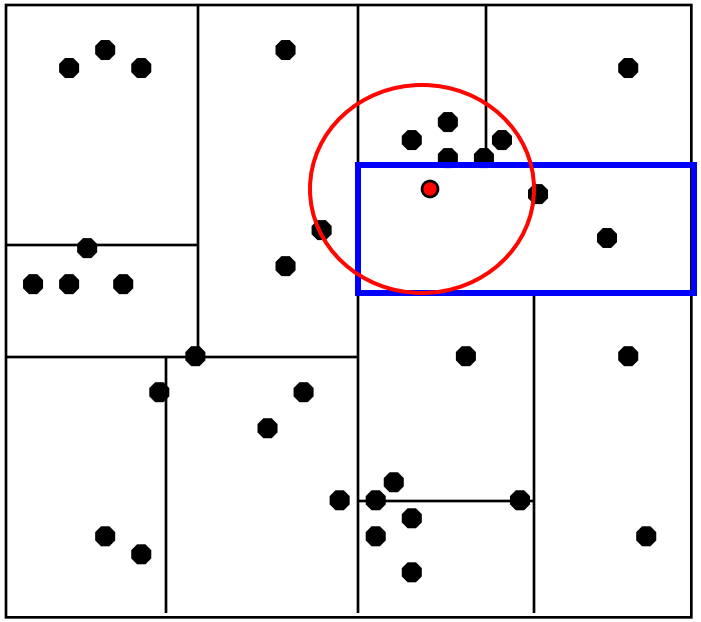
Nächster Nachbar mit k-d-Bäumen

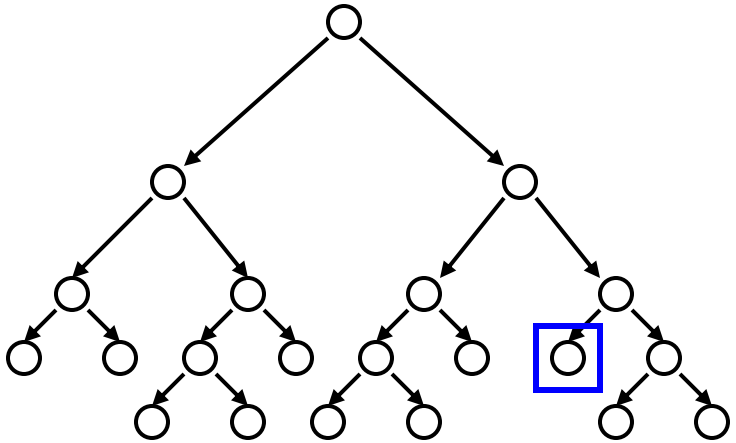




Wenn Blattknoten erreicht,
berechne Distanz zu jedem Punkt im Knoten

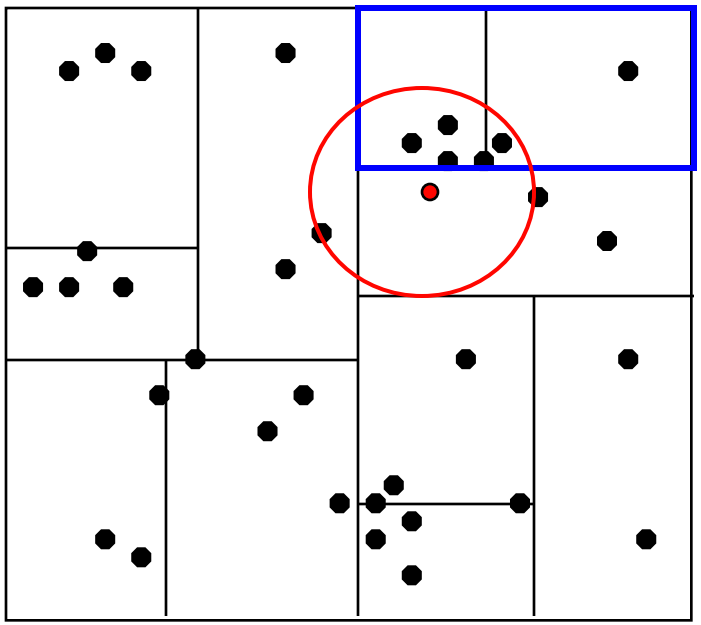
Nächster Nachbar mit k-d-Bäumen

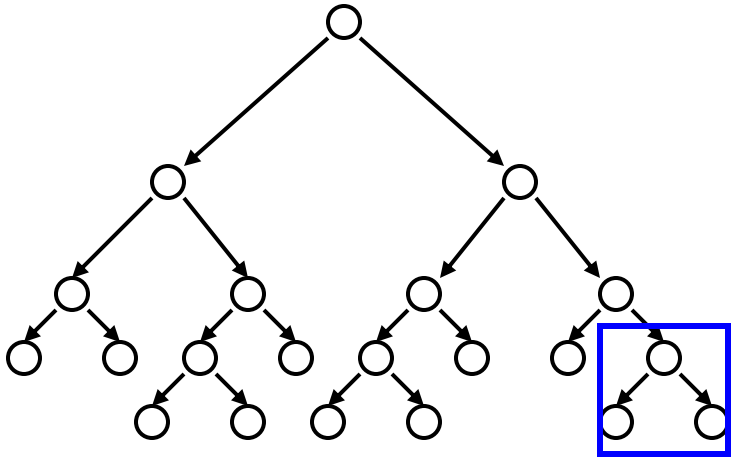




Bestimme Distanzgrenze (rot)

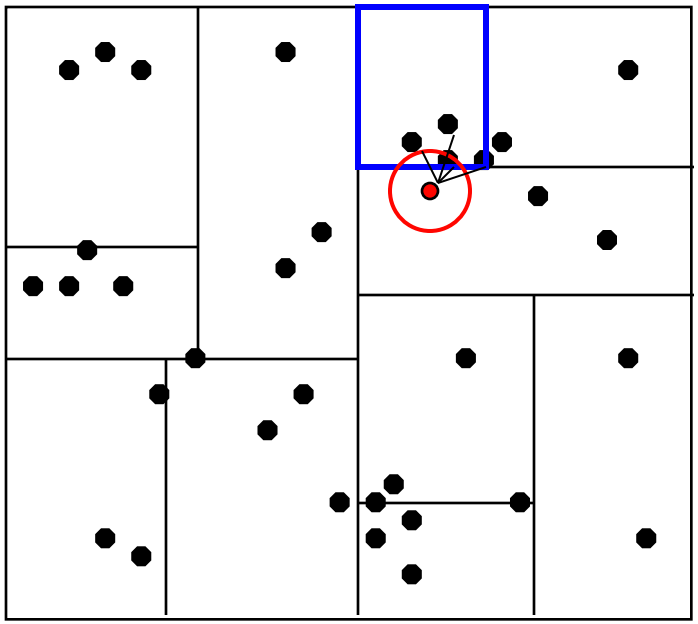
Nächster Nachbar mit k-d-Bäumen

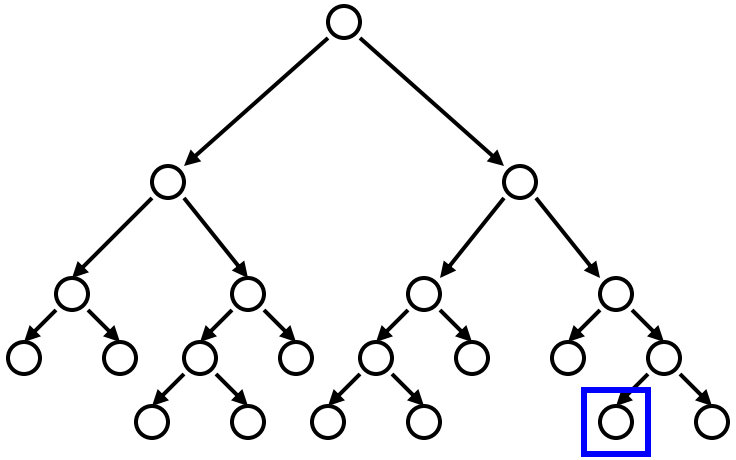




Backtracking: Versuche anderen Zweig bei jedem erreichten Knoten

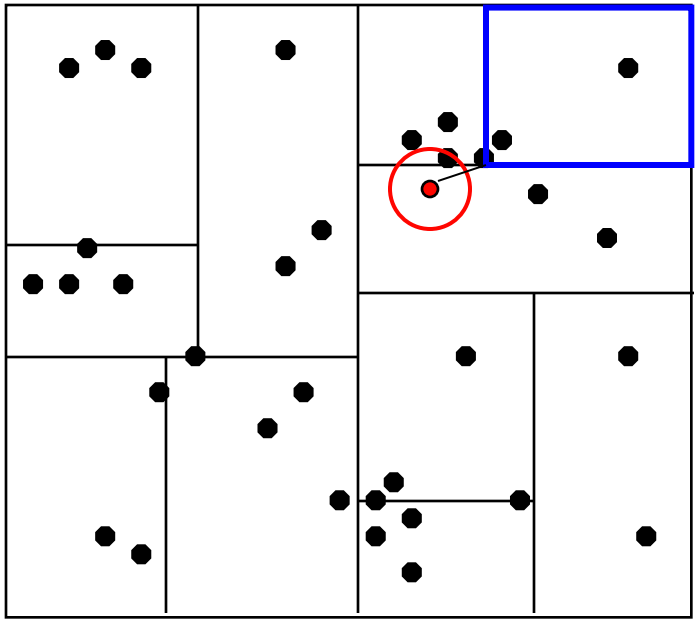
Nächster Nachbar mit k-d-Bäumen

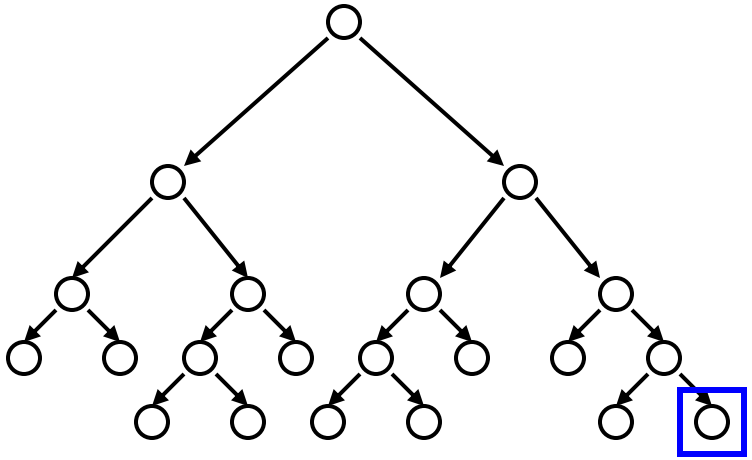




Wenn ein neuer Knoten gefunden wird,
kann Distanzgrenze neu bestimmt werden

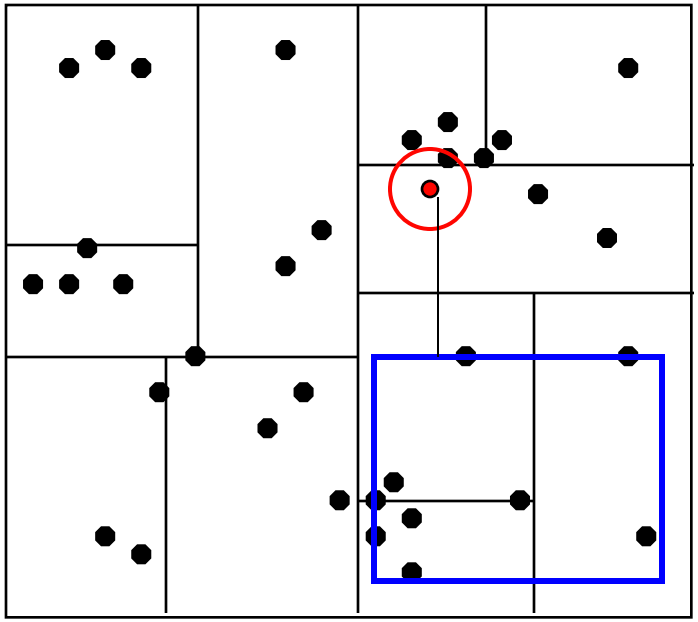
Nächster Nachbar mit k-d-Bäumen

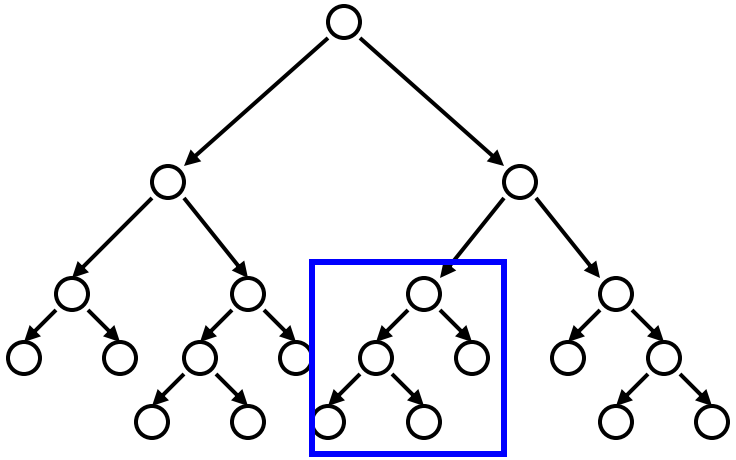




Mit der Distanzgrenze und der Grenze der Daten unter einem Knoten, können Teilbäume ausgeschlossen werden

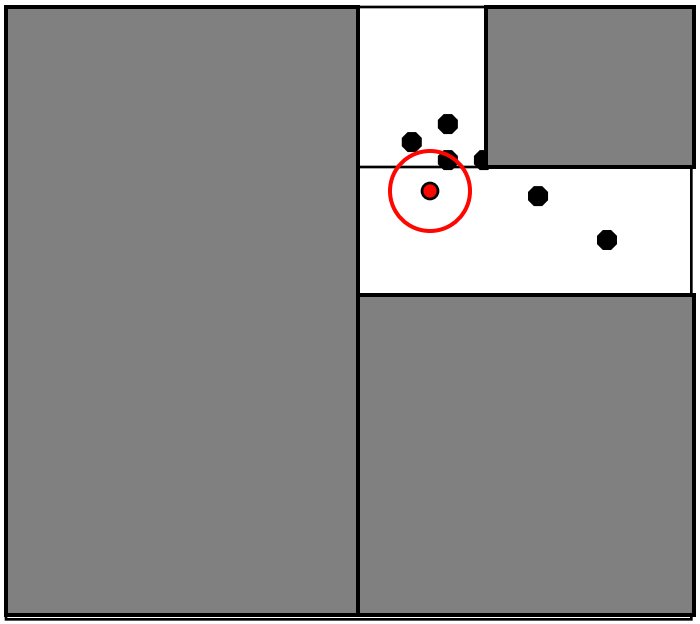
Nächster Nachbar mit k-d-Bäumen

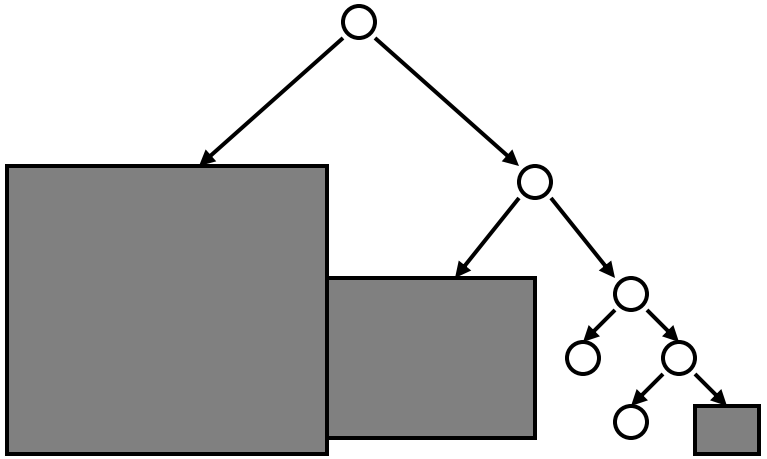




Mit der Distanzgrenze und der Grenze der Daten unter einem Knoten, können Teilbäume ausgeschlossen werden

Nächster Nachbar mit k-d-Bäumen





Mit der Distanzgrenze und der Grenze der Daten unter einem Knoten, können Teilbäume ausgeschlossen werden

Nächste-Nachbarn-Algorithmen

- Orchards Algorithmus (1991)
  - Verwendet $O(n^2)$ Speicher, ist aber schnell

    M.T. Orchard, A fast nearest-neighbor search algorithm, Proc. ICASSP'**91**

- Annulus Algorithmus
  - Ähnlich zu Orchard, verwendet O(n) Speicher nutzt aber wesentlich mehr Distanzberechnungen

    Clarkson, Kenneth L. "Nearest-neighbor searching and metric space dimensions." In: Nearest-neighbor methods for learning and vision: theory and practice: 15-59, **2006**

- Principal Component Partitioning (PCP)
  - Zatloukal, Johnson, Ladner (1999)
    Ähnlich zu k-d-Bäumen, schnell

    K. Zatloukal, R.E. Ladner and M.H. Johnson. Nearest Neighbor Search in Vector Quantization. Workshop on Algorithm Engineering and Experimentation, January **1999**

http://simsearch.yury.name/tutorial.html

CURE: Datenstrukturen und Komplexität

- kd-Baum, um Repräsentanten zu speichern:
  - `nn_kd_tree(p, T, max_dist, filter)`
    // max_dist und filter in obiges Verfahren leicht integrierbar
  - Vorverarbeitungszeit zum Aufbau: $O(n \cdot \log n)$
  - NN Anfragezeit: $O(\log n)$ bis $O(n)$ je nachdem ob Index wirksam oder nicht
  - Platzkomplexität: $O(n)$
- Min-Priority-Queue
- Im Vergleich zu einfachen hierarchischen Clusterverfahren bei CURE empirische Reduktion der Zeitkomplexität von $O(n^2 \cdot \log n)$ auf $O(n^2)$
- $O(n)$ Platzkomplexität gesamt

Laufzeit	Fib. Heap oder Brodal Queue	Radix-Heap
insert	O(1)	O(log C)
next	O(1)	O(1)
delete_next	O(log n)	O(1)
delete	O(log n)	O(1)
merge	O(1)	n/a
decrease_key	O(1)	O(1)

Clusterbildung: Übersicht

- Partitionierungsbasiert
  - K-Means
  - K-Medoids
  - BFR
  - DBSCAN
- Hierarchisch (zusammenballende Clusterbildung)
  - CURE
  - BIRCH
  - Spektrale Clusterbildung

BIRCH

Balanced Iterative Reducing and Clustering using Hierarchies

- Entwickelt für sehr große Datenmengen
  - Zeit und Speicher stark begrenzt (lineare IO-Kosten)
  - Gesamtdatensatz muss nicht a priori vorliegen
  - Inkrementelle und dynamische Clusterbildung für eingehende Objekte
  - Nur einmalige Betrachtung von Daten
- Zwei Phasen:
  - Scan der DB, um In-Memory-Baum aufzubauen
  - Clustering-Algorithmus, um Blattknoten zusammenzufassen

Zhang, T.; Ramakrishnan, R.; Livny, M. (1996). "BIRCH: An Efficient Data Clustering Method for Very Large Databases". Proc. ACM SIGMOD International Conference on Management of Data,. pp. 103–114, **1996**

Beschreibung von Clustern

Für ein Cluster von Punkten $\{\vec{X}_i\}$ definieren wir:

Zentroid: $\vec{X0} = \frac{\sum_{i=1}^{N} \vec{X}_i}{N}$

Radius (emp. Varianz): mittlere Distanz zum Zentroiden

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{X}_i - \vec{X0})^2}{N}\right)^{\frac{1}{2}}$$

Durchmesser: mittlere paarweise Distanz in einem Cluster

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X}_i - \vec{X}_j)^2}{N(N-1)}\right)^{\frac{1}{2}}$$

Durchmesser und Radius alternativ verwendet

Alternative Ähnlichkeitsmetriken (Cluster-Cluster)

Euklidische Zentroidendistanz: $D0 = ((\vec{X}0_1 - \vec{X}0_2)^2)^{\frac{1}{2}}$

Manhattan Zentroidendistanz: $D1 = |\vec{X}0_1 - \vec{X}0_2| = \sum_{i=1}^{d} |\vec{X}0_1^{(i)} - \vec{X}0_1^{(i)}|$

Mittlere Inter-Cluster-Dist.: $D2 = (\frac{\sum_{i=1}^{N_1} \sum_{j=N_1+1}^{N_1+N_2} (\vec{X}_i - \vec{X}_j)^2}{N_1 N_2})^{\frac{1}{2}}$

Mittlere Intra-Cluster-Dist.: $D3 = (\frac{\sum_{i=1}^{N_1+N_2} \sum_{j=1}^{N_1+N_2} (\vec{X}_i - \vec{X}_j)^2}{(N_1+N_2)(N_1+N_2-1)})^{\frac{1}{2}}$

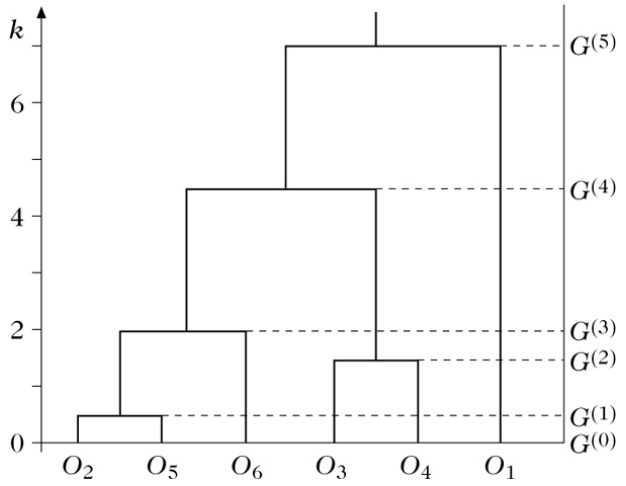
„Varianzanstieg“ (Ward):

$$D4 = (\sum_{k=1}^{N_1+N_2} (\vec{X}_k - \frac{\sum_{l=1}^{N_1+N_2} \vec{X}_l}{N_1+N_2})^2 - \sum_{i=1}^{N_1} (\vec{X}_i - \frac{\sum_{l=1}^{N_1} \vec{X}_l}{N_1})^2 - \sum_{j=N_1+1}^{N_1+N_2} (\vec{X}_j - \frac{\sum_{l=N_1+1}^{N_1+N_2} \vec{X}_l}{N_2})^2)^{\frac{1}{2}}$$

Bei kategorialen Variablen andere Distanzmaße

Clusterbeschreibung

- BIRCH baut beim Lesen der Daten ein Dendrogramm, genannt Cluster-Funktions-Baum (CF-Baum) auf
- Eintrag im CF-Baum repräsentiert Cluster von Objekten



  - charakterisiert durch 3-Tupel (N, LS, SS): Cluster-Funktion
  - N ist die Anzahl der der Objekte im Cluster $\{\mathbf{P}_i\}_{i \in N}$
  - LS (linear sum) und SS (square sum) sind wie folgt definiert: $LS = \sum_{i=1}^{N} \mathbf{P}_i$

$$SS = \sum_{i=1}^{N} (\mathbf{P}_i)^2$$ Komponentenweise Quadrierung

$\mathbf{P}_i$ sind d-dimensionale Punkte, dito LS und SS

Eigenschaften von CF

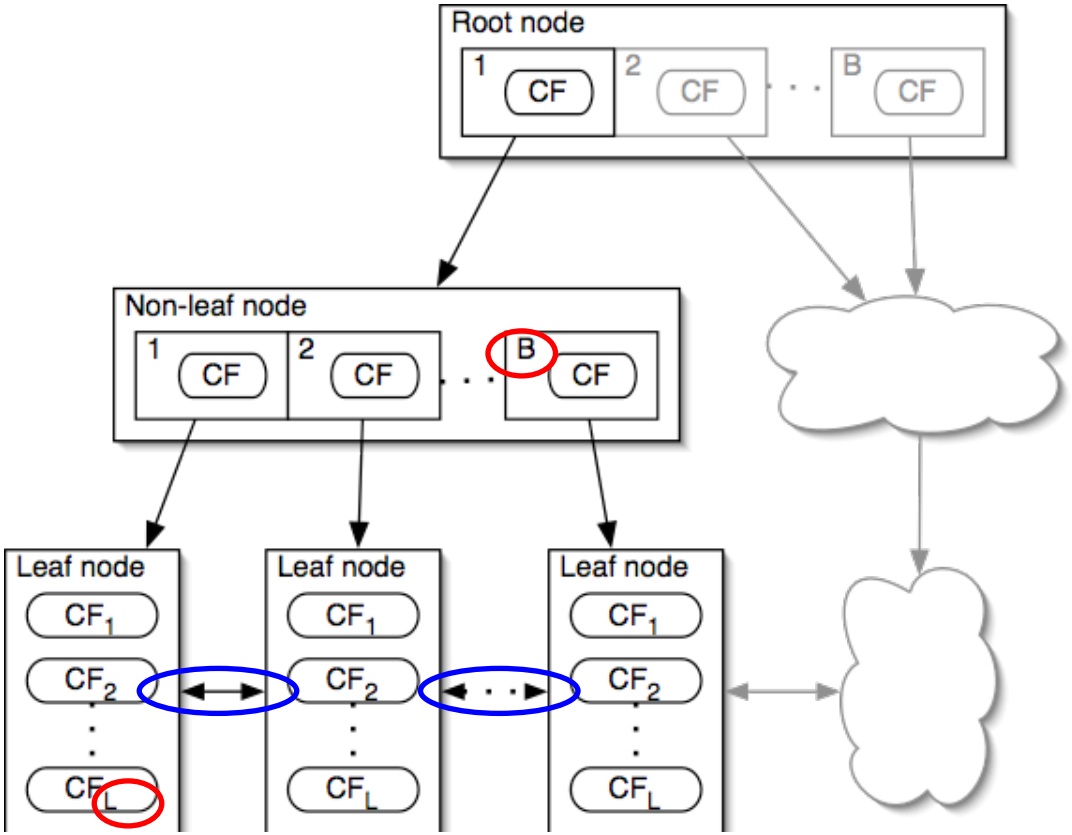
- CF-Einträge sind kompakt
  - Signifikant weniger Speicher nötig als wenn man alle Datenpunkte der Untercluster speichern würde
- Ein CF-Eintrag enthält genügend Info zur Berechnung von D0-D4 (Lance/Williams Formel)
- Additivitätstheorem gestattet, Untercluster inkrementell und konsistent zusammenzufassen (für disjunktes CF1, CF2):

$$\mathbf{CF_1} + \mathbf{CF_2} = (N_1 + N_2, L\vec{S}_1 + L\vec{S}_2, SS_1 + SS_2)$$

- CFs können gemäß Di sortiert werden

CF-Baum (vgl. B-Baum)

- Jeder Nicht-Blatt-Knoten hat höchstens B CF-Einträge (CF_i, $child_i$)
- Jeder Blattknoten hat höchstens L CF-Einträge
- *Durchmesser* oder alternativ *Radius* eines jeden Blattclusters < *Schwellwert* T
- Knotengröße durch die Dimension des Datenraums und durch Inputparameter P (Seitengröße) festgelegt



B = Verzweigungsfaktor

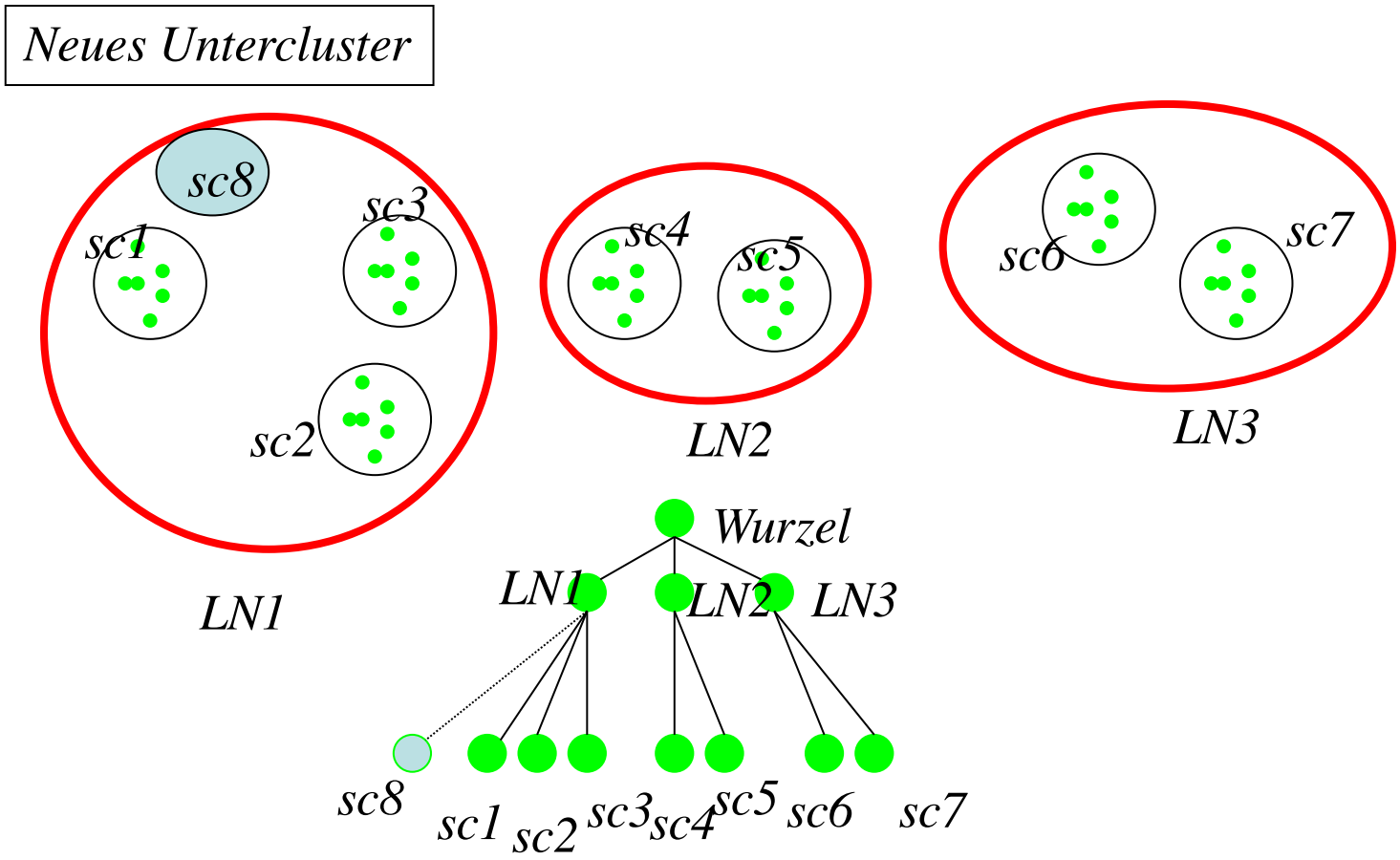
Einfügeoperation im CF-Baum

- Steige hinab von Wurzel, bis richtiger Blattknoten gefunden
  - Folge dem ”D_i-nächster"-CF-Pfad für Di = D0 / … / D4
- Modifiziere den Blattknoten
  - Integration in alten Cluster, wenn Durchmesser oder Radius < T, sonst neuen Cluster beginnen
  - Falls das “D_i-nächste”-CF-Blatt voll, erstelle neuen CF-Eintrag. Falls kein Platz für neuen Blattknoten, splitte Elternknoten
- Traversiere zurück (wie beim B-Baum)
  - und aktualisiere dabei CFs auf dem Pfad oder spalte Knoten auf

CF-Baum-Umstrukturierung

- Falls kein Speicherplatz, heb Schwellwert T an
  - Dadurch können CFs mehr Daten aufnehmen
- Umstrukturierung durch höhere Ts erlauben es, CFs zu gruppieren
- Reduzierbarkeitstheorem
  - Anhebung von T resultiert in kleinerem CF-Baum
  - Umstrukturierung benötigt höchstens h extra Speicherseiten (h ist die Höhe des CF-Baums, abhängig von T)

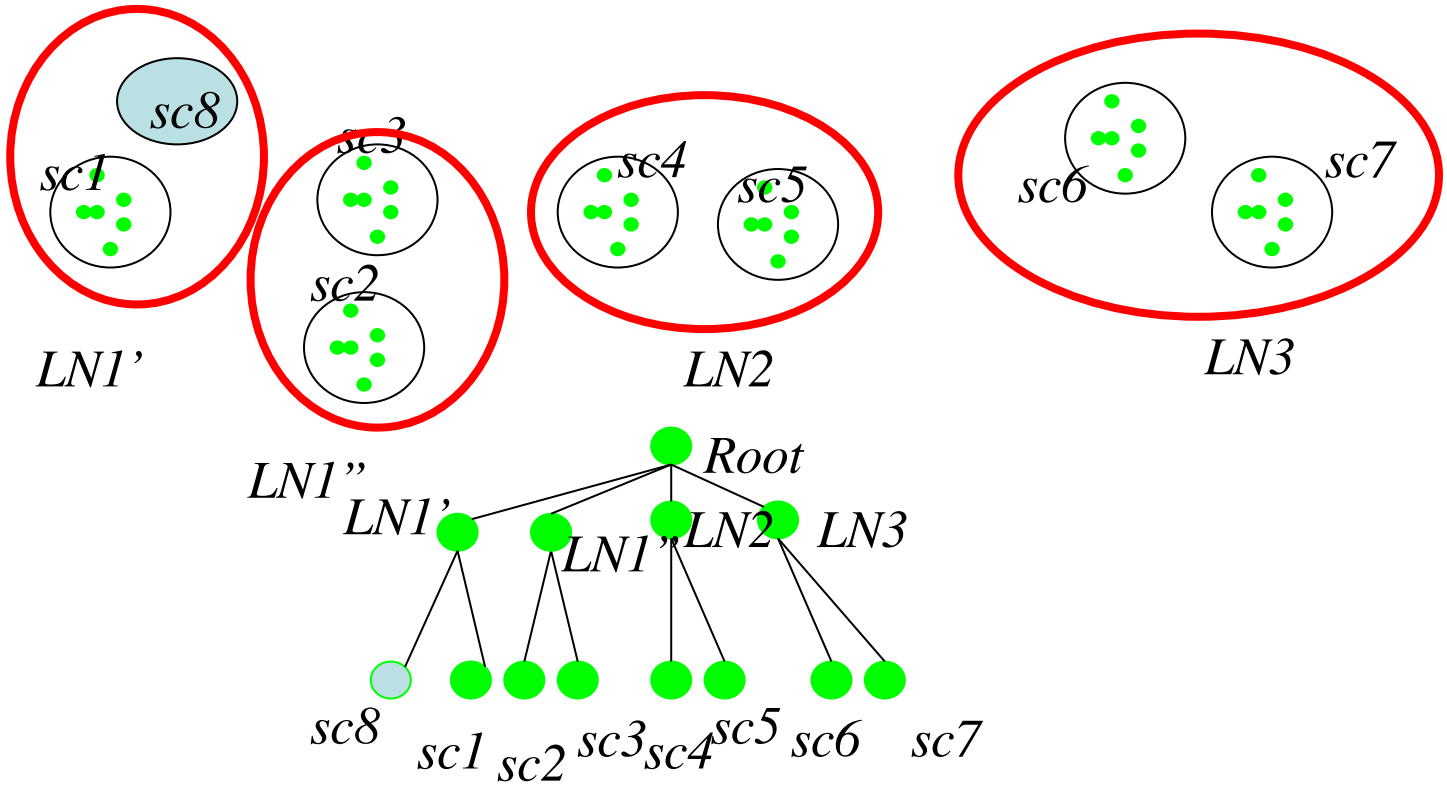
BIRCH: Beispiel



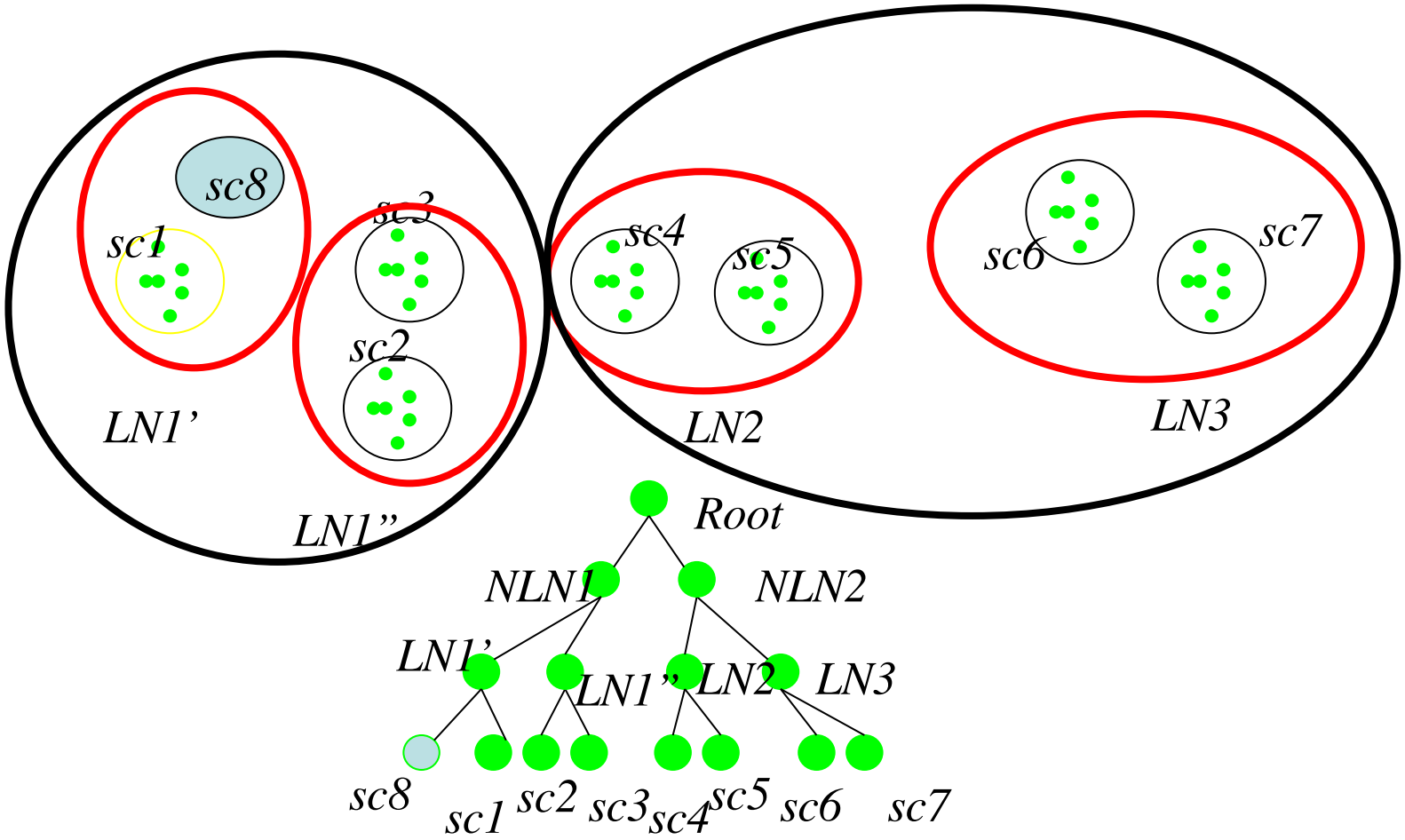
sc = Subcluster, ln = leaf node

Einfüge-Operation in BIRCH

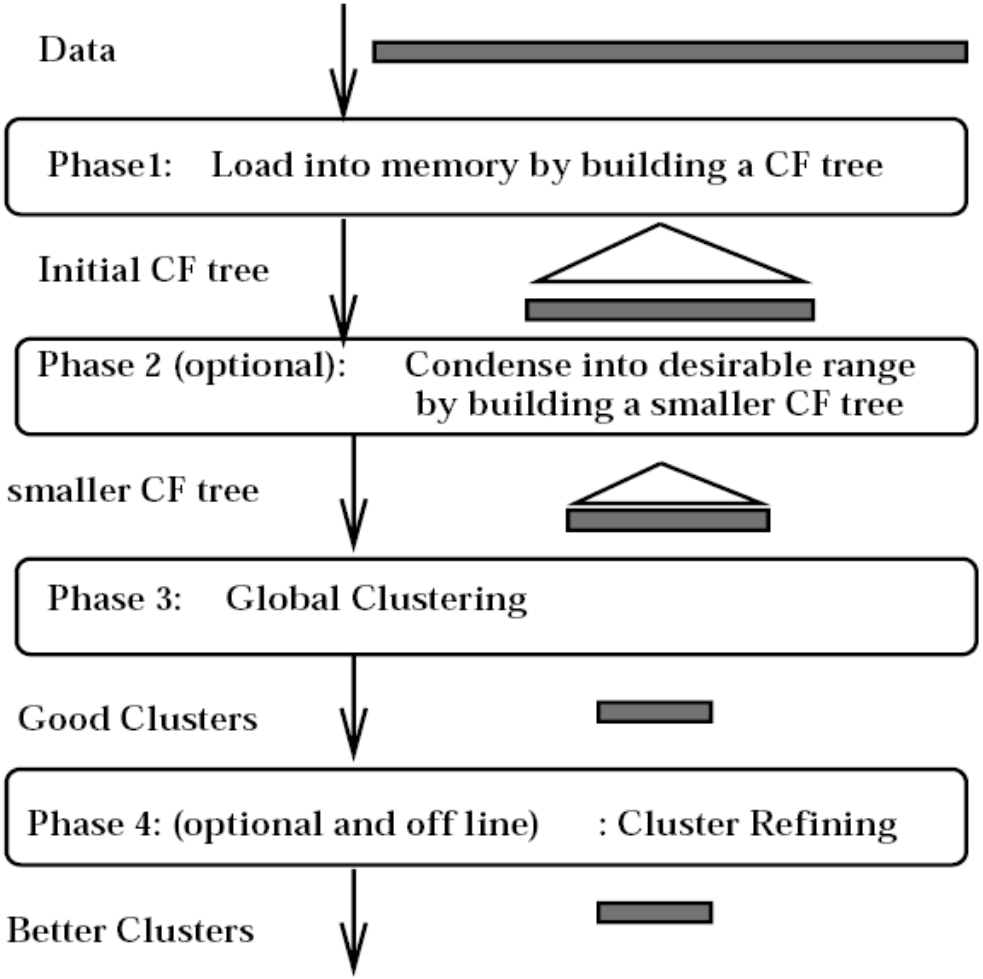
Falls der Verzweigungsfaktor eines Blattes (LN Knoten) 3 nicht übersteigen kann, wird LN1 aufgespalten (Beachte: die „Blätter“ im folgenden Baum sind nicht die Blattknoten, sondern Inhalte der Blattknoten, und stellen die eigentlichen Cluster dar



Falls der Verzweigungsfaktor eines Nicht-Blattes 3 nicht übersteigen kann, wird die Wurzel aufgespalten und die Höhe des CF-Baums steigt um 1



BIRCH Übersicht



Ggf. durch Anheben von T

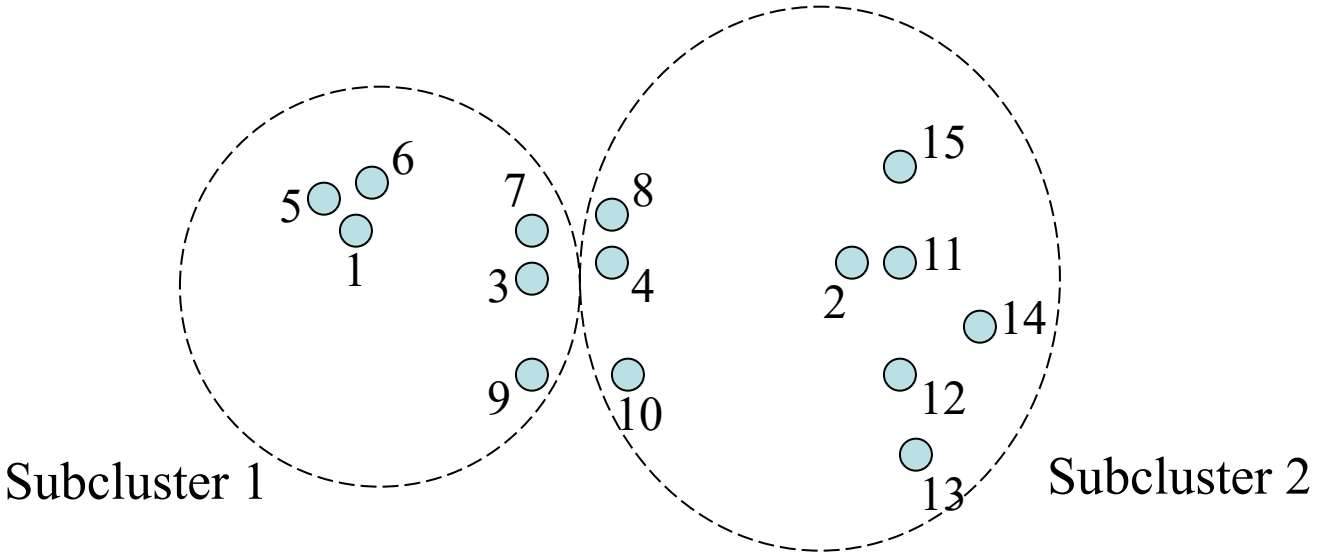
Verkleinern und Anpassen des CF-Baumes

Wende zusammenballendes Clusterverfahren auf Blattknoten an

Verbessere Cluster

Problematische Fälle für BIRCH

- Die Objekte sind gemäß der Ankunftsreihenfolge nummeriert; nimm an, dass der Abstand zwischen Objekt 1 und 2 den (Durchmesser-)Schwellwert überschreitet.



Abhängigkeit von der Reihenfolge

- Inkrementelle Algorithmen wie z.B. BIRCH sind von der Reihenfolge abhängig
- Split- und Merge-Operation können die Reihenfolgeabhängigkeit zu einem gewissen Grad ausgleichen
- Dennoch kann die Reihenfolgeabhängigkeit nicht komplett vermieden werden

Probleme mit dem CF-Baum

- Anzahl der Einträge in CF-Knoten durch Seitengröße beschränkt. Daher stellt ein CF-Knoten eventuell keinen natürlichen Cluster dar.
  - Zwei Unter-Cluster, die einem Cluster untergeordnet sein sollten, sind auf zwei Knoten verteilt.
  - Zwei Unter-Cluster, die nicht einem Cluster untergeordnet sein sollten, werde auf einem Knoten gehalten (abhängig von der Eingabereihenfolge und der Datenverzerrung)

Probleme mit dem CF Baum (Forts.)

- Sensitivität bzgl. Eingabereihenfolge
  - Datenpunkt kann in Blattknoten landen, wo es nicht hingehört (z.B. wegen Durchmesserschwellwert)
  - Falls Datenpunkt zweimal eingefügt wird (zu verschiedenen Zeiten), kann es in zwei Kopien in unterschiedlichen Blättern landen

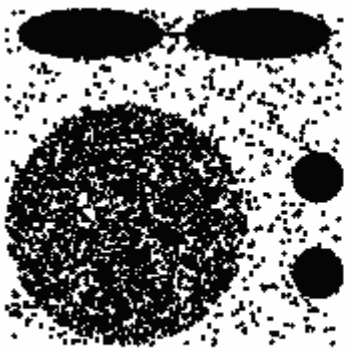
Experimente

- BIRCH
- CURE
  - Partitionierung in k = 5 Cluster
  - Stichprobengroße = 2.5% der initialen Datensatzgröße
- MST („minimum spanning tree“ flächenüberdeckend)
  - Wenn $\alpha = 0$, wird CURE zu MST

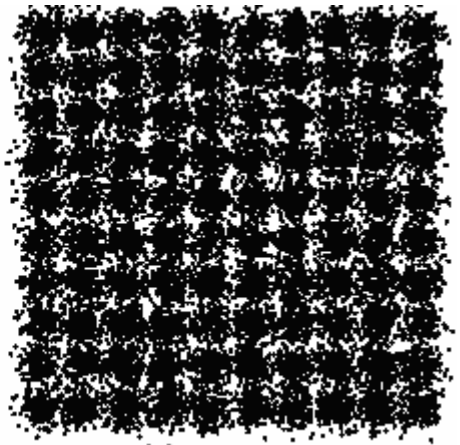
Experimente: Datensätze

Experiment mit 2-d Datensätzen

- Datensatz 1 enthält einen großen Kreis, zwei Ellipsen und zwei kleine Datensätze
- Datensatz 2 besteht aus 100 Clustern im Gitter (Datenpunkte je Cluster normalverteilt um Zentrum)

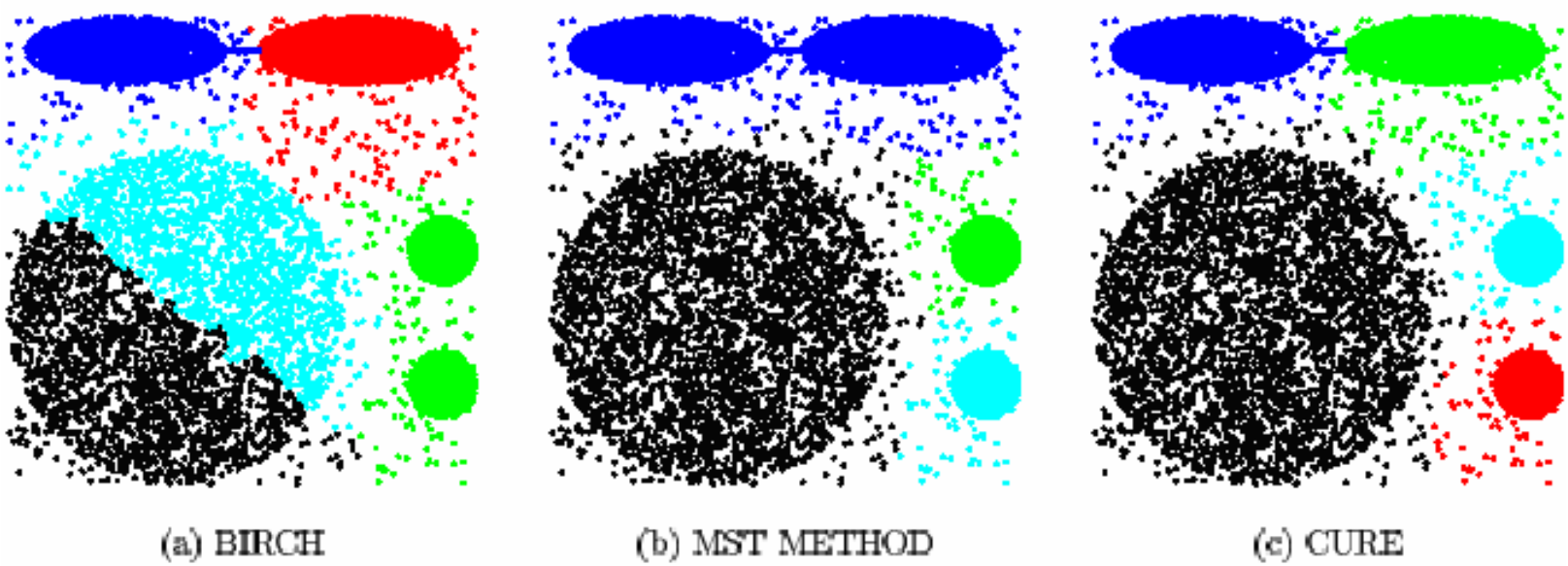


(a) Data set 1



(b) Data set 2

Experimente: Qualität der Clusterbildung

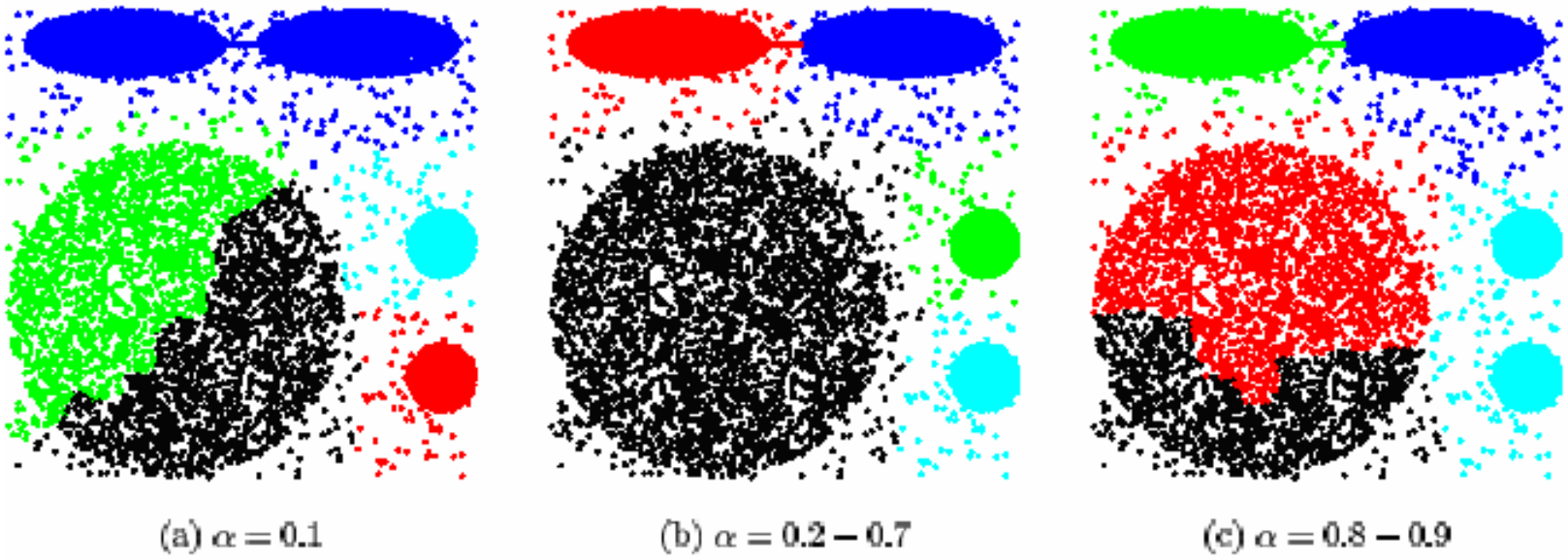


(a) BIRCH (b) MST METHOD (c) CURE

Experimente: Parametersensitivität

CURE-Schrumpfungsfaktor α:

- $0.2 - 0.7$ ist ein guter Bereich für α

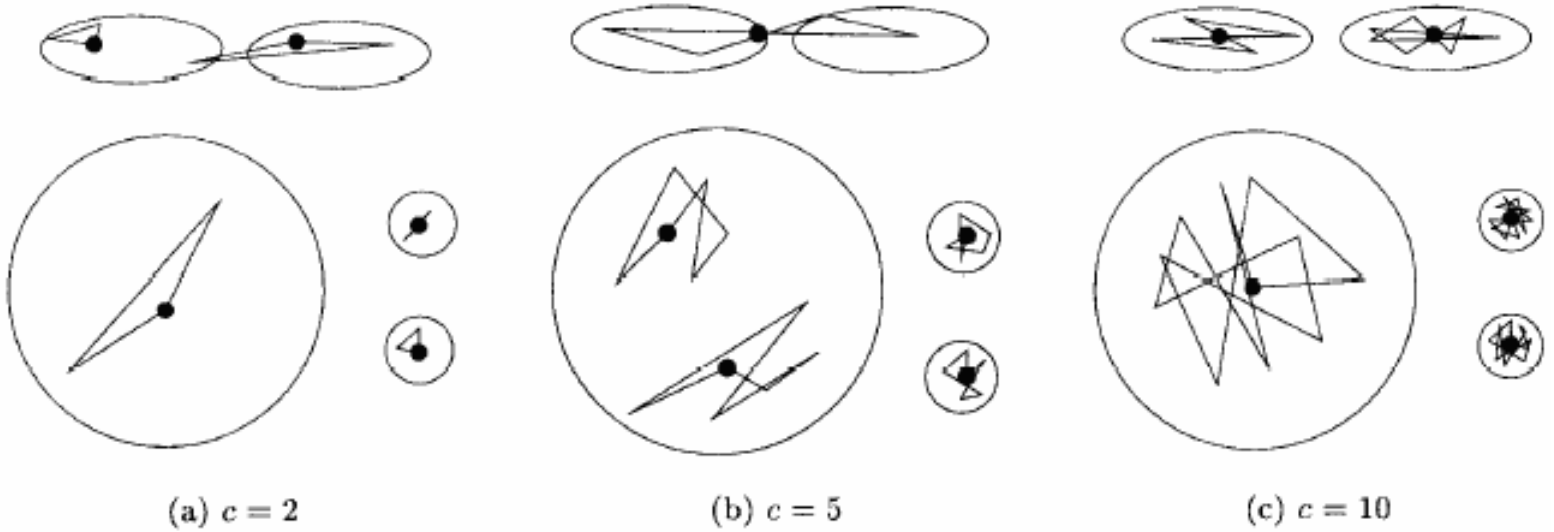


(a) $\alpha = 0.1$ (b) $\alpha = 0.2 - 0.7$ (c) $\alpha = 0.8 - 0.9$

Experimente: #Repräsentanten

Anzahl der Repräsentanten c:

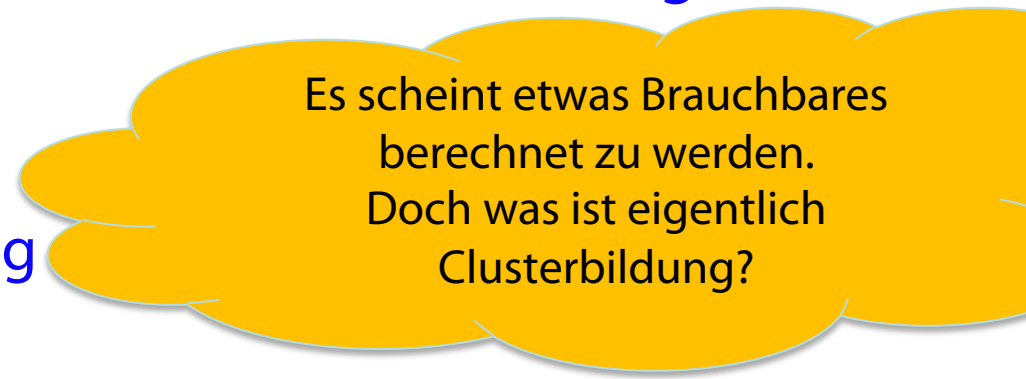
- Für kleine Werte von c leidet die Qualität der Clusterbildung
- Mit größerem c arbeitet CURE recht gut (aber langsamer)



(a) $c = 2$ (b) $c = 5$ (c) $c = 10$

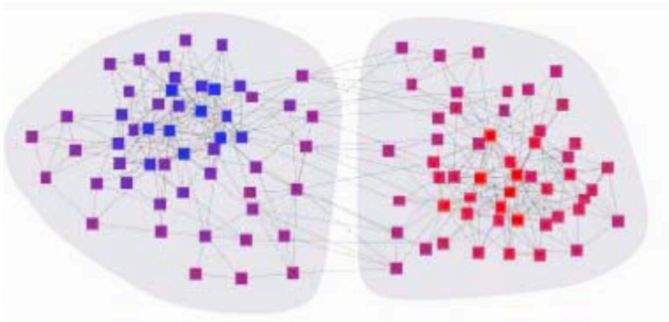
Clusterbildung: Übersicht

- Partitionierungsbasiert
  - K-Means
  - K-Medoids
  - BFR
  - DBSCAN
- Hierarchisch (zusammenballende Clusterbildung)
  - CURE
  - BIRCH
  - Spektrale Clusterbildung



Spektrale Clusterbildung

- Zentrale Idee: Repräsentiere Datenpunkte als Knoten V eines Graphen G = (V, E)
- Die Kanten E sind gewichtet: Hohes Gewicht bedeutet, dass die verbundenen Knoten sehr „ähnlich“ sind

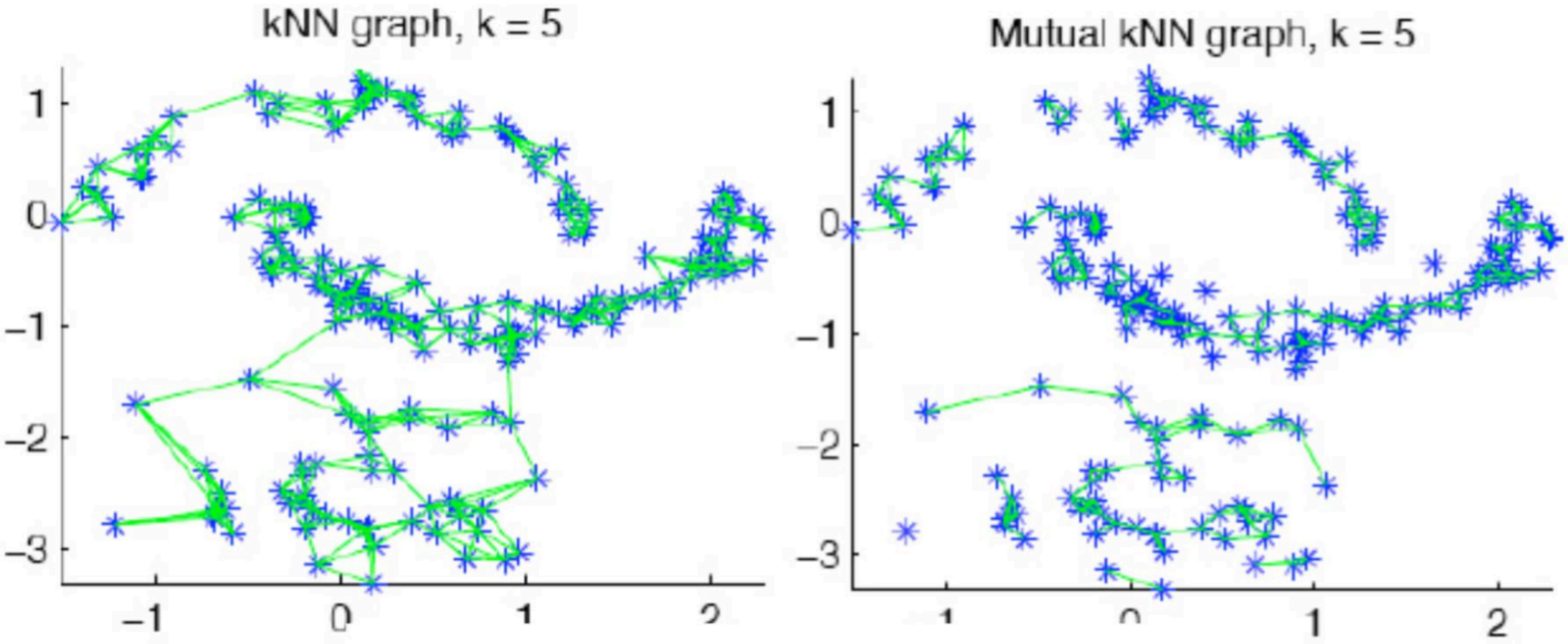


- Graph beschrieben durch Gewichtsmatrix W
- Verwende Spektrum der Gewichtsmatrix zur Clusterbildung (Spektrum = Menge/Vektor der verschiedenen Eigenwerte)

Erzeugung des Graphen auf Punktmenge M

- Vollständige Verbindung
- Nächste-Nachbarn-Graph: Distanz von p zu q nicht größer als von p zu irgendeinem anderen Objekt in M
- K-Nächste-Nachbarn-Graph (k-NN)
  - Distanz von p zu q unter den k-kleinsten Distanzen von p zu anderen Objekten in M
- Wechselseitiger K-Nächste-Nachbarn-Graph
  - K-NN-Graph mit k als maximaler Grad:
  - $E(v_i, v_j)$ gdw v_j zu den k-nächsten von v_i gehört und umgekehrt auch v_i zu den k-nächsten von v_j gehört.
- ε-Nachbarschaft (Umgebung mit Radius ε)

Notiz: K-NN Nicht symmetrisch; in Darstellungen Richtung meist vergessen


kNN graph, k = 5
Mutual kNN graph, k = 5
epsilon–graph, epsilon=0.3

Bestimmung der Gewichte

- Für Gewichte z.B. Gauß-Kern:

Für $i \neq j$

$$W(i,j) = \exp \frac{-\|x_i - x_j\|_2^2}{\sigma^2}$$

Sonst $W(i,j) = 0$

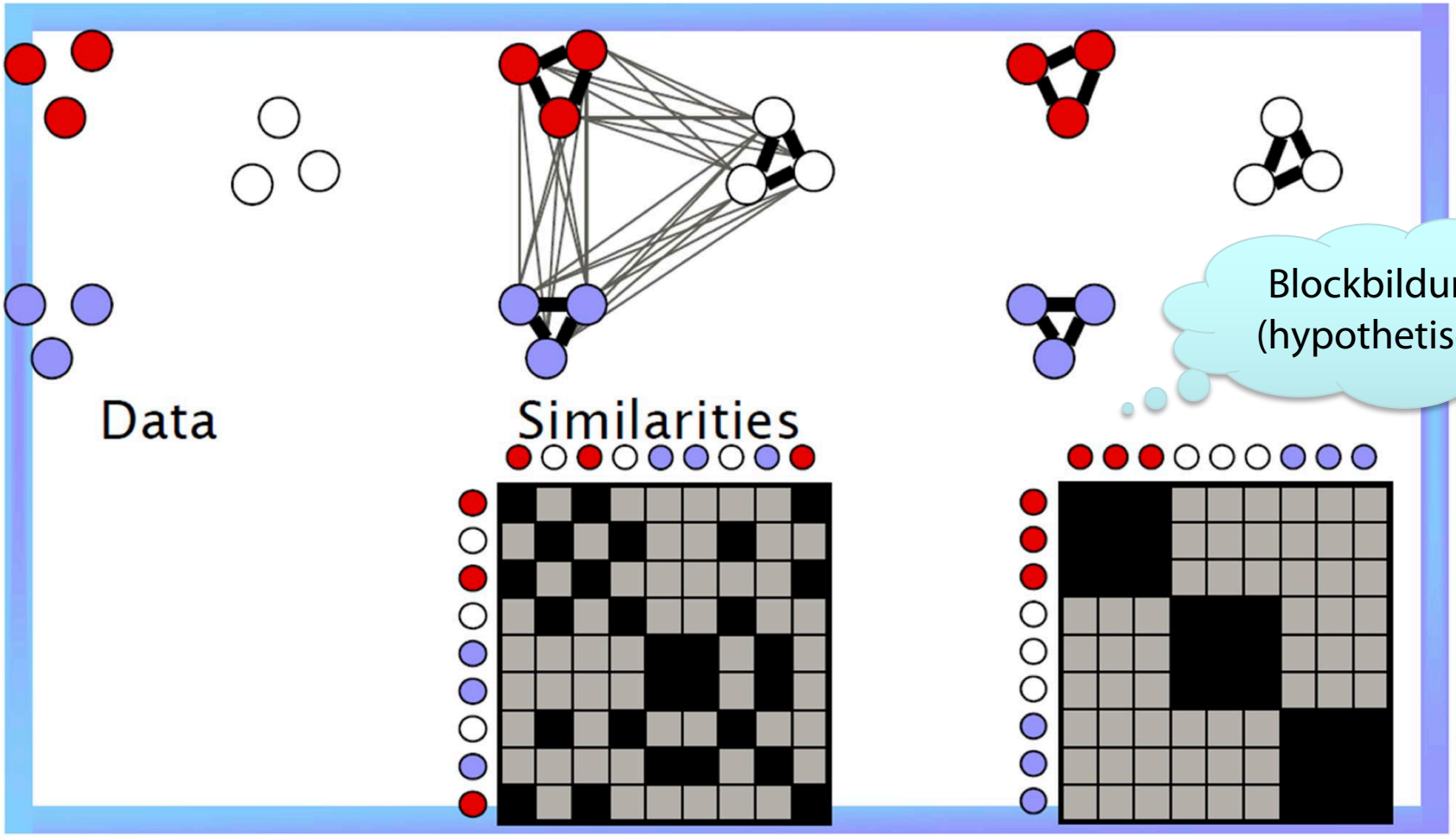
σ^2 ist ein Skalierungsparameter

- Zur Erinnerung:
  - Normalverteilung

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad -\infty < x < \infty$$

  - $\mu := 0$

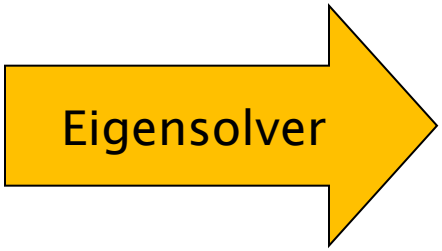
Graphpartitionierung



Eigenvektoren und Blöcke

- Block-Matrizen haben Block-Eigenvektoren

1	1	0	0
1	1	0	0
0	0	1	1
0	0	1	1

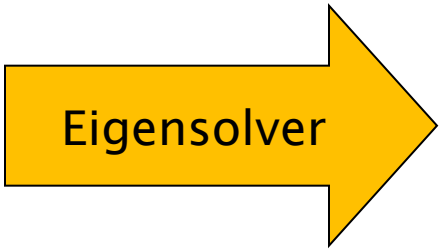


$\lambda_1 = 2$ $\lambda_2 = 2$ $\lambda_3 = 0$ $\lambda_4 = 0$

.71	0
.71	0
0	.71
0	.71

- Fast-Block-Matrizen haben Fast-Block-Eigenvektoren

1	1	.2	0
1	1	0	-.2
.2	0	1	1
0	-.2	1	1



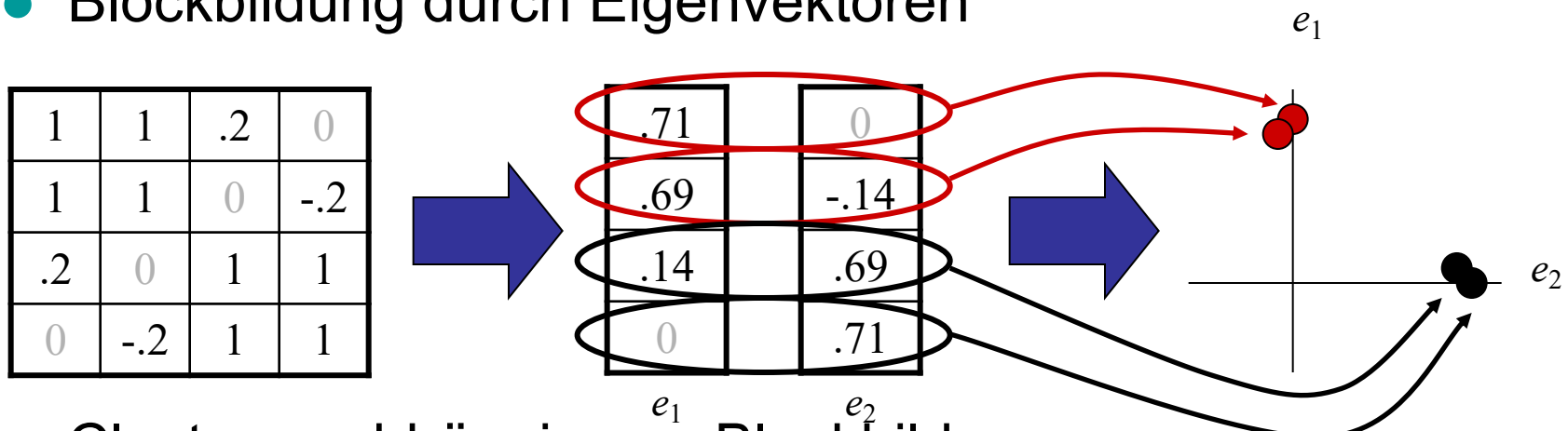
$\lambda_1 = 2.02$ $\lambda_2 = 2.02$ $\lambda_3 = -0.02$ $\lambda_4 = -0.02$

.71	0
.69	-.14
.14	.69
0	.71

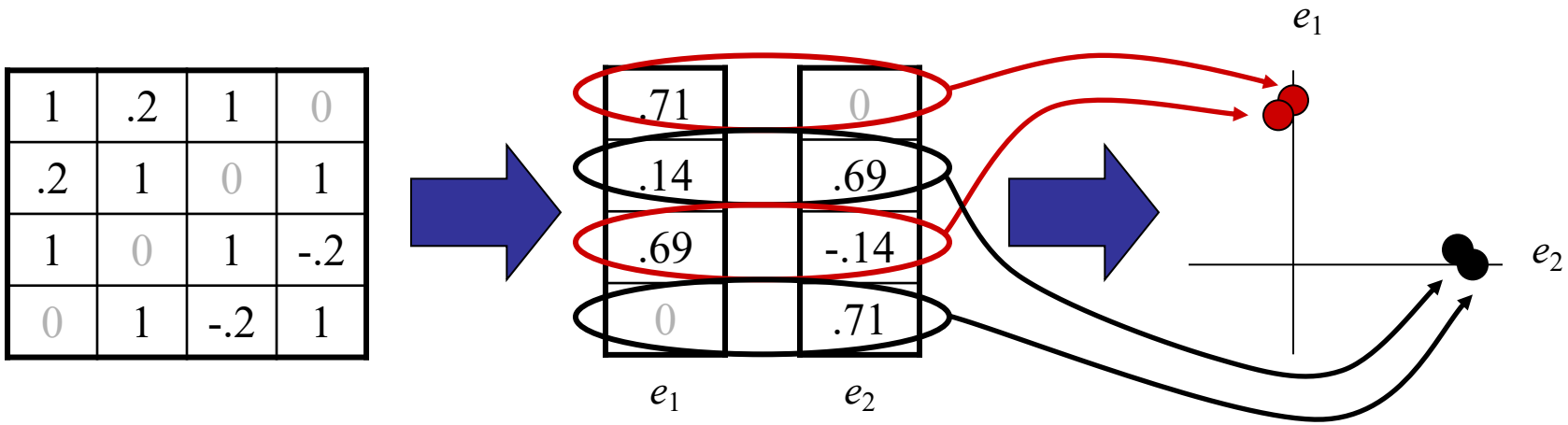
* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University

Spektraler Raum

- Blockbildung durch Eigenvektoren



- Cluster unabhängig von Blockbildung



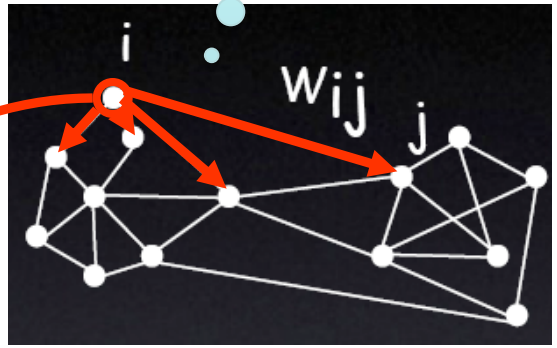
* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University

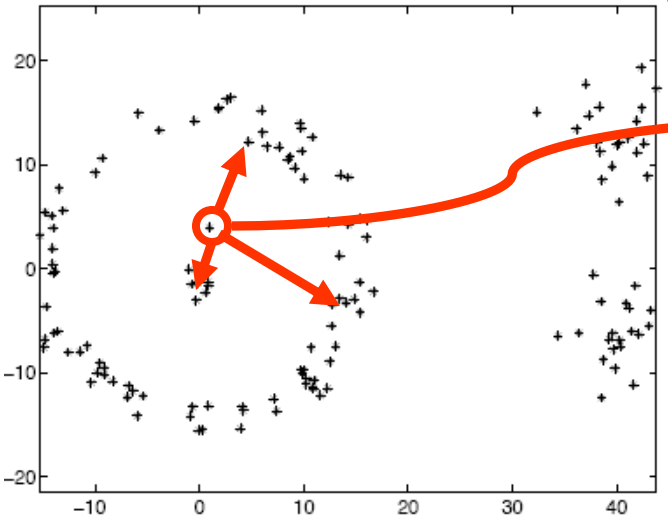
Graphterminologie

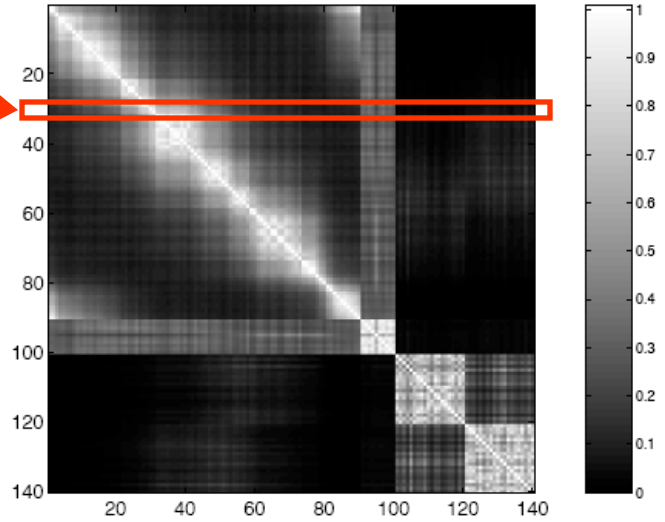
Der Übersicht halber nur einige Punkte eingezeichnet

- Ähnlichkeitsmatrix $W = [w_{i,j}]$

$$w_{i,j} = e^{\frac{-\|X_{(i)} - X_{(j)}\|_2^2}{\sigma_X^2}}$$

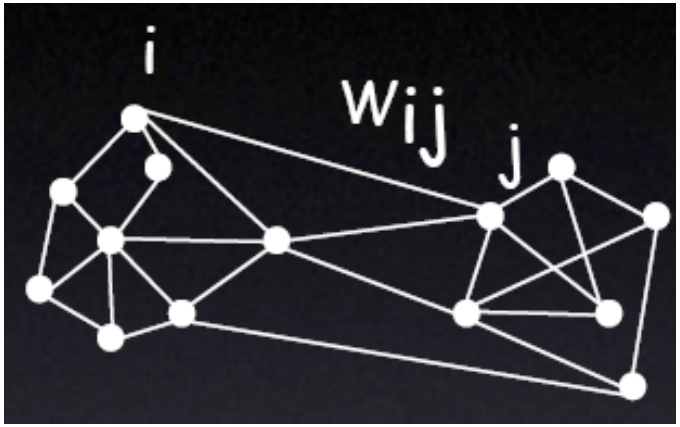


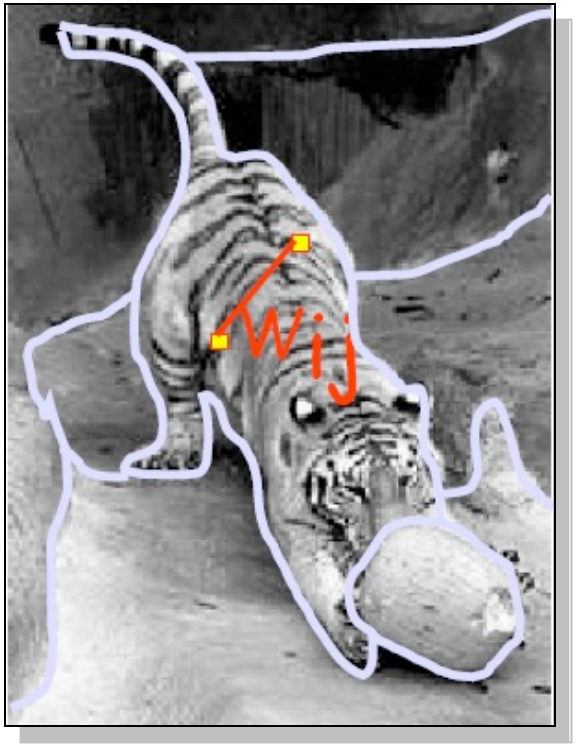




Slides from Jianbo Shi

Anwendung: Graph-basierte Bildsegmentierung





Pixel
Pixel-Ähnlichkeit

Slides from Jianbo Shi

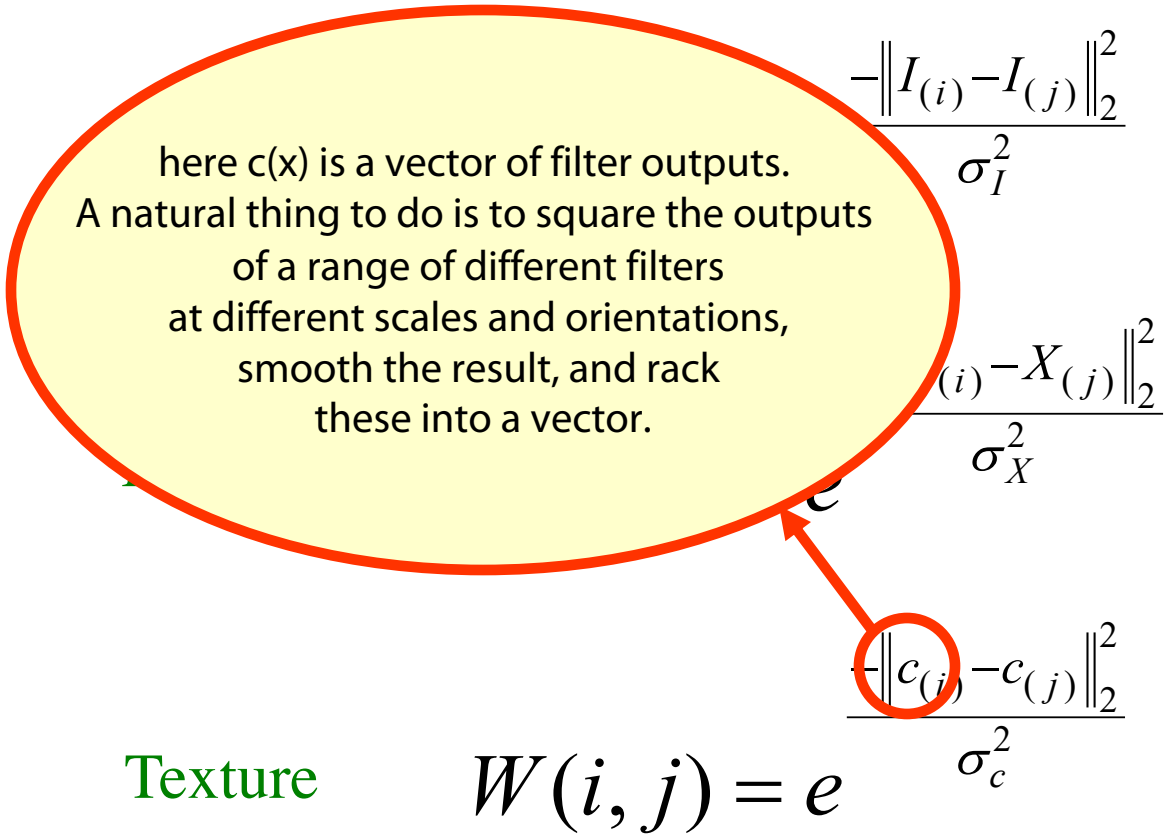
Pixelähnlichkeitsmaße

Intensity $$W(i,j) = e^{\frac{-\left\|I_{(i)} - I_{(j)}\right\|_2^2}{\sigma_I^2}}$$

Distance $$W(i,j) = e^{\frac{-\left\|X_{(i)} - X_{(j)}\right\|_2^2}{\sigma_X^2}}$$

Texture $$W(i,j) = e^{\frac{-\left\|c_{(i)} - c_{(j)}\right\|_2^2}{\sigma_c^2}}$$

Pixel similarity functions



Pixel similarity functions

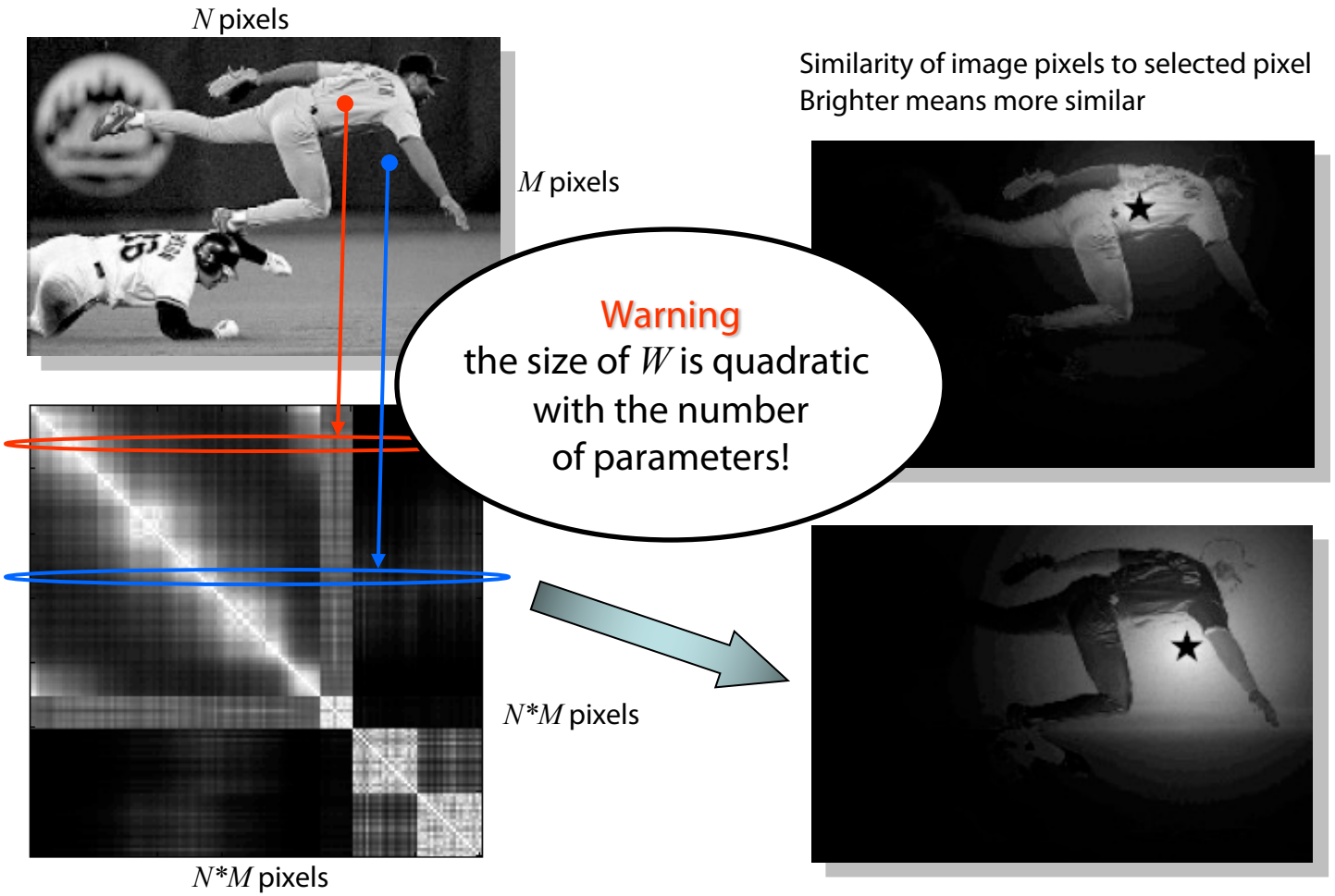
Oder verwende eine Kombination…

Intensity $$W(i,j) = e^{\frac{-\left\|I_{(i)} - I_{(j)}\right\|_2^2}{\sigma_I^2}}$$

Distance $$W(i,j) = e^{\frac{-\left\|X_{(i)} - X_{(j)}\right\|_2^2}{\sigma_X^2}}$$

Texture $$W(i,j) = e^{\frac{-\left\|c_{(i)} - c_{(j)}\right\|_2^2}{\sigma_c^2}}$$

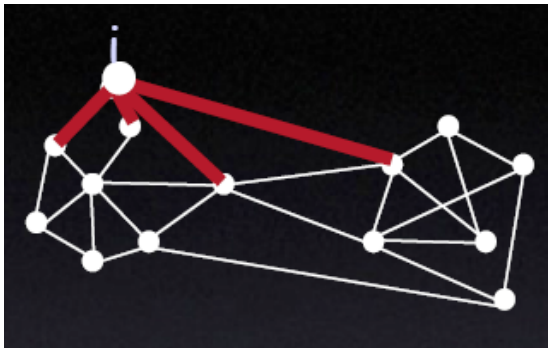
Ähnlichkeitsmatrix (Affinitätsmatrix)

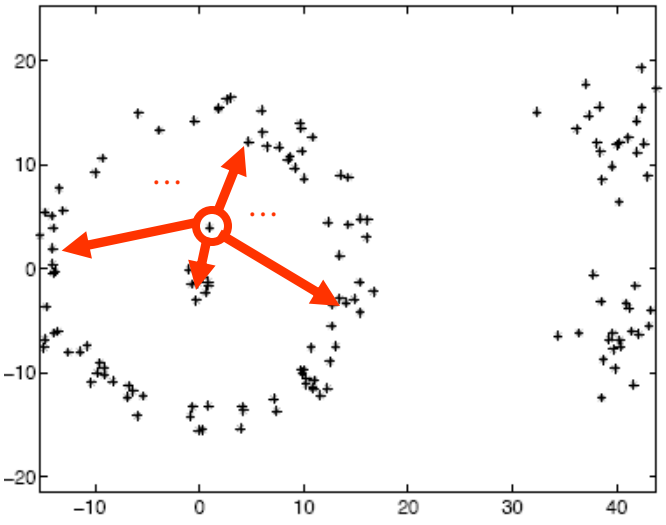


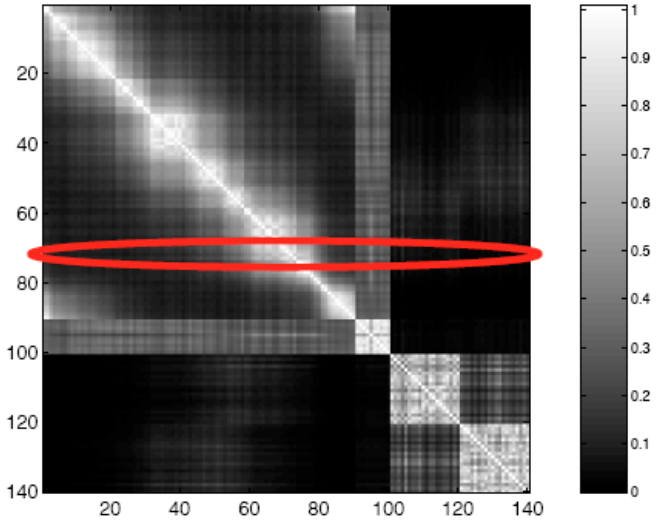
Graphterminologie

- Grad eines Knotens

$$d_i = \sum_j w_{i,j}$$





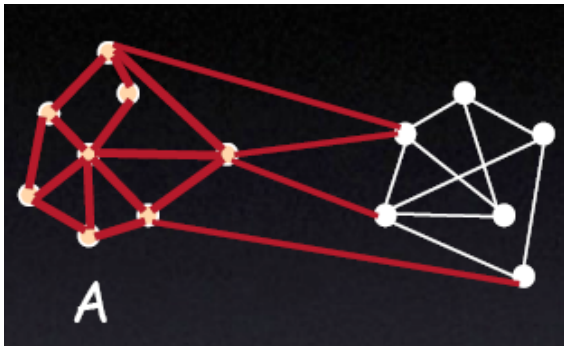


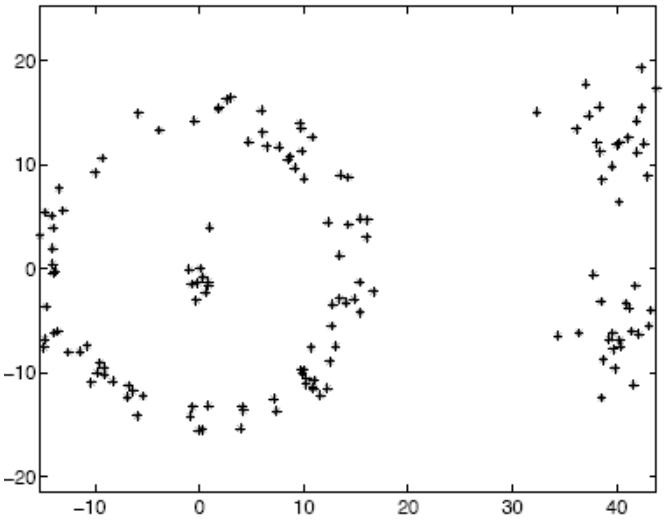
Slides from Jianbo Shi

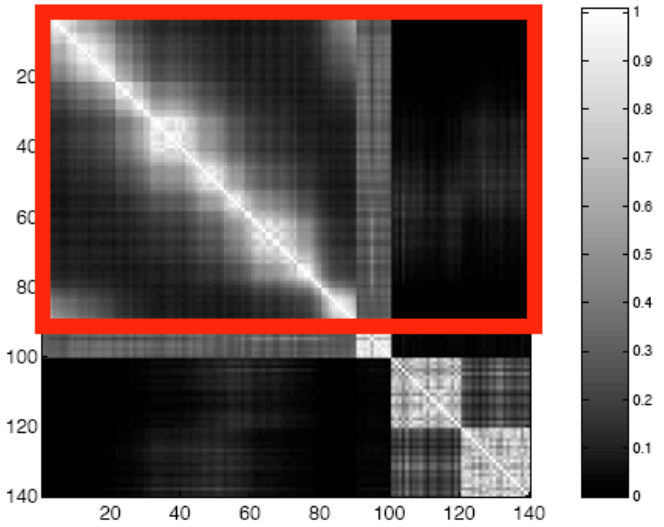
Graphterminologie

- Volumen einer Knotenmenge

$$vol(A) = \sum_{i \in A} d_i, A \subseteq V$$





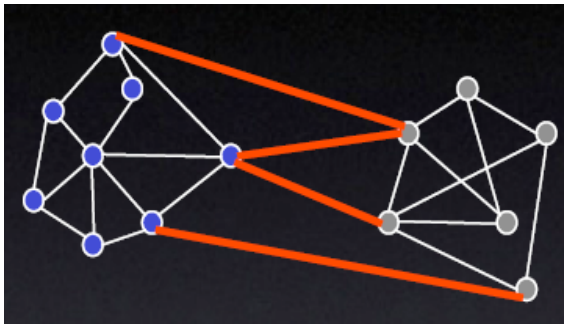


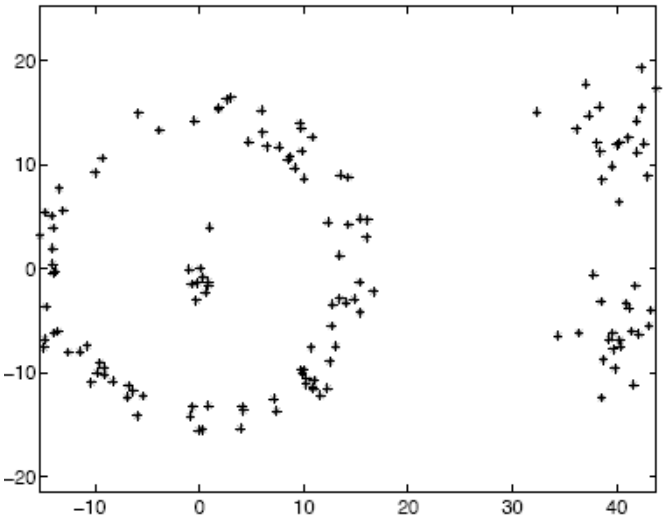
Slides from Jianbo Shi

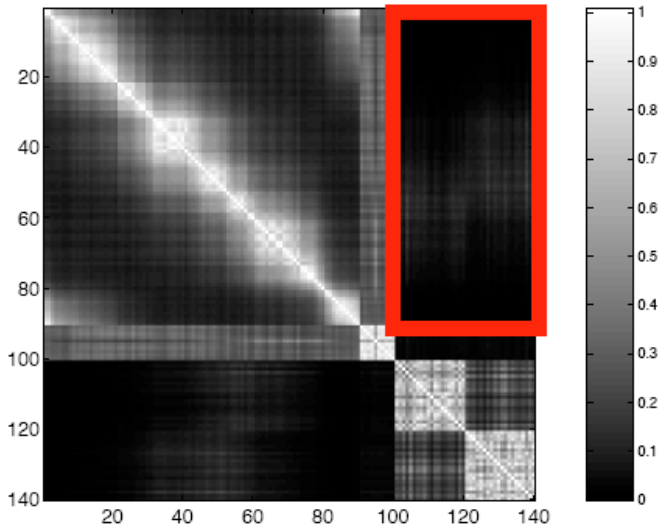
Graphterminologie

- Schnitte in einem Graph:

$$cut(A, \overline{A}) = \sum_{i \in A, j \in \overline{A}} w_{i,j}$$







Slides from Jianbo Shi

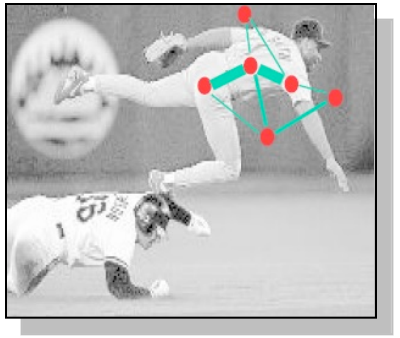
Repräsentation

Partition matrix X:

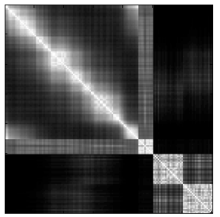
$$X = [X_1, \ldots, X_K]$$

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} \text{pixels} \end{matrix}$$

segments



Pair-wise similarity matrix W: $W(i,j) = aff(i,j)$

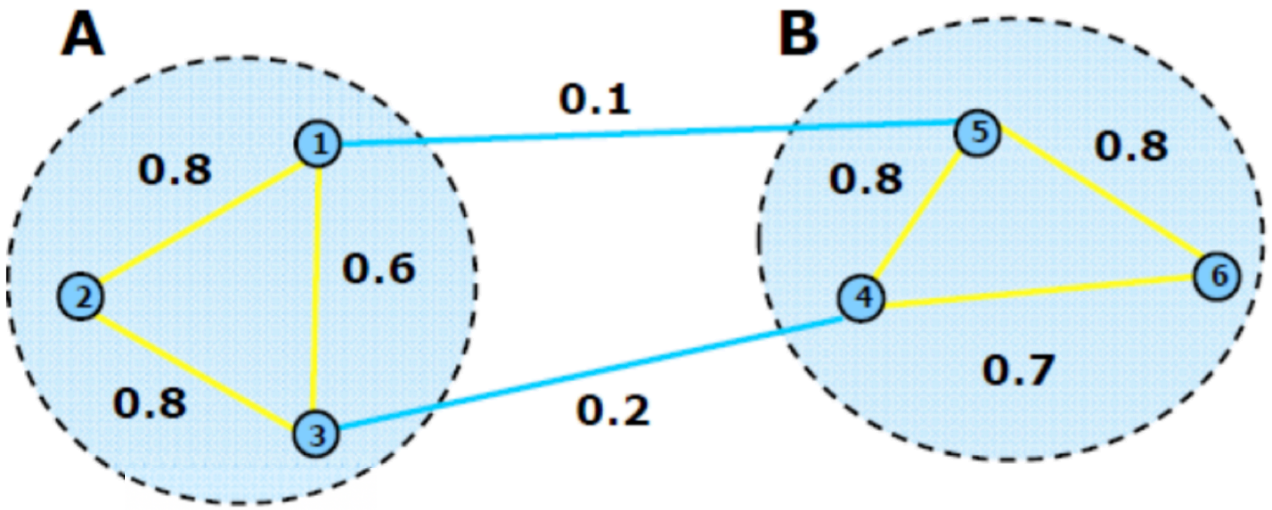


Degree matrix D: $D(i,i) = \sum_j w_{i,j}$

Laplacian matrix L: $L = D - W$

Gewichteter Schnitt

- Betrachte Partition von G in Teile A und B



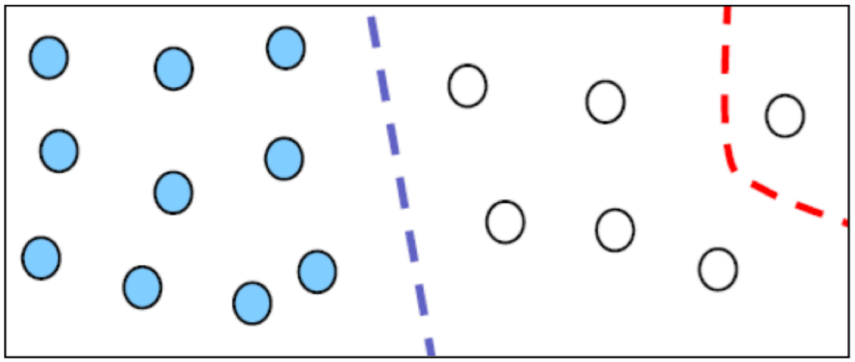
- Gew. Schnitt(A, B): Summe der Gewichte der Menge der Kanten, die die Partitionen verbinden $cut(A,B) = \sum_{i \in A, j \in B} w_{ij} = 0.3$
- Ziel: Finde (gute) Partitionierung mit einem minimalen gewichteten Schnitt

Min-Schnitt-Zielfunktion

- Minimiere Summe der Gewichte der Verbindungen zwischen Partitionen A und B

$$\min_{A \cap B = \emptyset, A \cup B = V} Cut(A, B)$$

- Problem: Degenerierte Lösungen präferiert



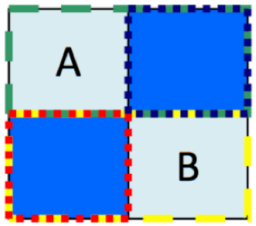
- Gesucht: Zielfunktion, die balancierte Lösungen präferiert

Daniel Delling, Daniel Fleischman, Andrew V Goldberg, Ilya Razenshteyn, and Renato F Werneck. An exact combinatorial algorithm for minimum graph bisection. Mathematical Programming, 153(2):417–458, **2015**

Normalisierter Schnitt: Ncut

- Betrachte Verbindungsstruktur zwischen Partitionen relativ zu dem Volumen jeder Partition

$$Ncut(A,B) = \frac{cut(A,B)}{Vol(A)} + \frac{cut(A,B)}{Vol(B)}$$



$$Ncut(A,B) = cut(A,B)\frac{Vol(A)+Vol(B)}{Vol(A)Vol(B)}$$

- Maximal, wenn Vol(A) und Vol(B) gleich
- Folge: balancierte Partitionierung

J. Shi, J. Malik: Normalized cuts and image segmentation. In: IEEE Transactions on Pattern Analysis and Machine Intelligence. 22(8), S. 888–905, **2000**

Annahme (vorläufig)

- Gewichte sind alle gleich 1
- Wir wollen den Graphen mit n Knoten in zwei Teile zerlegen, so dass der Kantenschnitt minimal wird (Graph-Bisektion)
- Wir geben jedem Knoten v_i eine Nummer und verwenden einen Indikatorvektor x mit

$$x_i = \begin{cases} 1 & \text{falls } v_i \in V_1 \\ -1 & \text{falls } v_i \in V_2 \end{cases}$$

- Wir schreiben $\arg\min_{x \in \{+1,-1\}^n} Ncut(x)$
- Die Teile sollen gleich groß sein: $x^T \mathbf{1} = 0$

Bestimmung des Cuts unter der Annahme

- Falls v_i und v_j in derselben Partition liegen, dann gilt $x_i - x_j = 0$.
- Umgekehrt, falls v_i und v_j in unterschiedlichen Partitionen liegen, dann gilt auf Grund der unterschiedlichen Vorzeichen $(x_i - x_j)^2 = 4$.

$$
\begin{aligned}
Ncut(x) &= \frac{1}{4} \sum_{(i,j)\in E} (x_i - x_j)^2 \\
&= \frac{1}{4} \left(\sum_{(i,j)\in E} (x_i^2 + x_j^2) - \sum_{(i,j)\in E} 2x_i x_j \right) \\
&= \frac{1}{4} \left(x^T D x - x^T A x \right) \\
&= \frac{1}{4} x^T L x
\end{aligned}
$$

$$
- \sum_{(i,j)\in \mathcal{E}} 2x_i x_j = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} x_i A_{ij} x_j = x^T A x
$$

$$
\sum_{(i,j)\in \mathcal{E}} (x_i^2 + x_j^2) = \sum_{(i,j)\in \mathcal{E}} 2 = 2|\mathcal{E}| = \sum_{i=1}^{n} \deg(v_i) = x^T D\, x
$$

Bestimmung des Cuts unter der Annahme

- Formulierung des Problems

$$\begin{aligned} \text{Minimiere } f(x) &= \frac{1}{4} x^T L x \\ \text{s.t.} \quad x &\in \{+1, -1\}^n \\ x^T \mathbf{1} &= 0 \end{aligned}$$

- Ganzzahligkeitsbedingung macht Problem NP-schwer
- Aufweichung: $x_i \in \{-sqrt(n), sqrt(n)\}$
  $x^T x = n$, mit n Anzahl der Knoten im Graphen

A. Pothen. Graph partitioning algorithms with applications to scientific computing. Parallel Numerical Algorithms, D. E. Keyes, A. Sameh, and V. Venkatakrishnan, eds., Kluwer, 1997, pp. 323–368., **1997**.

T.F. Chan, P. Ciarlet Jr, W.K. Szeto, On the Optimality of the median cut spectral bisection graph partitioning method, Technical Report, UCLA, **1993**.

Bestimmung des Cuts unter der Annahme

- Neue Formulierung

$$\min_{x\in\{+1,-1\}^n, x^T 1=0} f(x) = \min_{x\in\{+1,-1\}^n, x^T 1=0} \frac{1}{4} x^T L x$$

$$\geqslant \min_{x^T x=n, x^T 1=0} \frac{1}{4} x^T L x$$

$$= \frac{1}{4} u_2^T L u_2$$

$$= \frac{n}{4} \lambda_2$$

u_2 ist zweiter Eigenvektor

$$\frac{v^t A v}{v^t v} = \lambda$$

$$n = v^{\mathrm{T}} v$$

Mathematischer Exkurs

- Sei A eine symmetrische Matrix: $A = A^t$

  Dann heißt

$$R(A, v) = \frac{v^t A v}{v^t v}$$

  der Rayleigh-Quotient von A
  für das Eigenwertproblem $Av = \lambda v$

- Der Rayleigh-Quotient definiert eine Abbildung von Eigenvektoren auf Eigenwerte

$$\frac{v^t A v}{v^t v} = \lambda$$

- NB: Kantengewichte = 1

min Ncut-Bestimmung mit beliebigen Gewichten

- Bestimmung von

$$\min_x Ncut(x) \text{ über } \min_y \frac{y^T(D-W)y}{y^T Dy} \qquad y \in \mathbb{R}^N$$

- (ohne Herleitung)

Lösung von *min Ncut*

- Abschwächung in kontinuierliche Domäne durch Lösen des generalisierten Eigenwert-Problems

$$\min_y y^T(D-W)y \quad \text{mit} \quad y^T D y = 1$$

Gleich große Partitionen

- Lagrange-Funktional

$$L(y,\lambda) = y^T(D-W)y - \lambda(y^T D y - 1)$$

- Partielle Ableitungen bzgl. y und Nullstellenbestimmung

führt zu $(D-W)y = \lambda D y$

Generalisiertes Eigenwertproblem

- Beachte: $(D-W)1 = 0$

  y = 1 ist der erste Eigenvektor mit Eigenwert 0

- 2. Eigenvektor von L liefert reelle Lösung von *min Ncut*

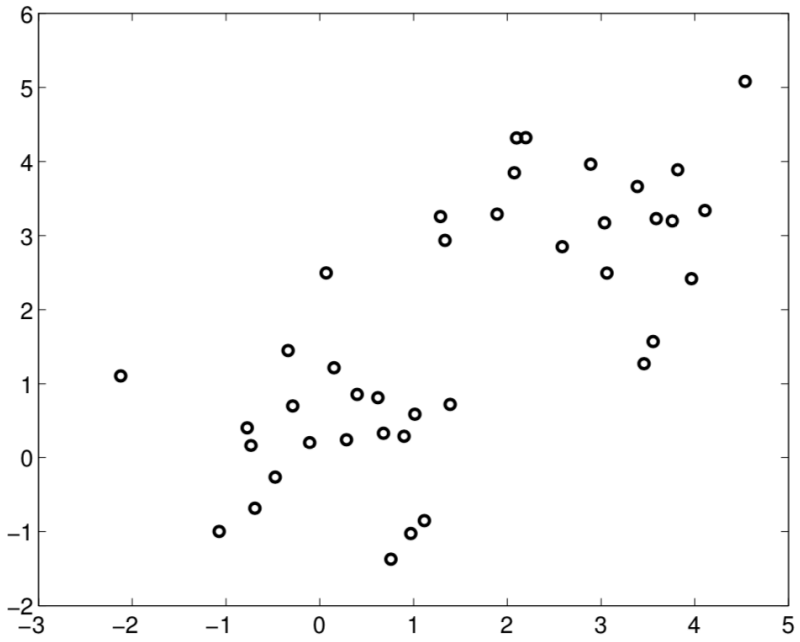
Mathematischer Exkurs

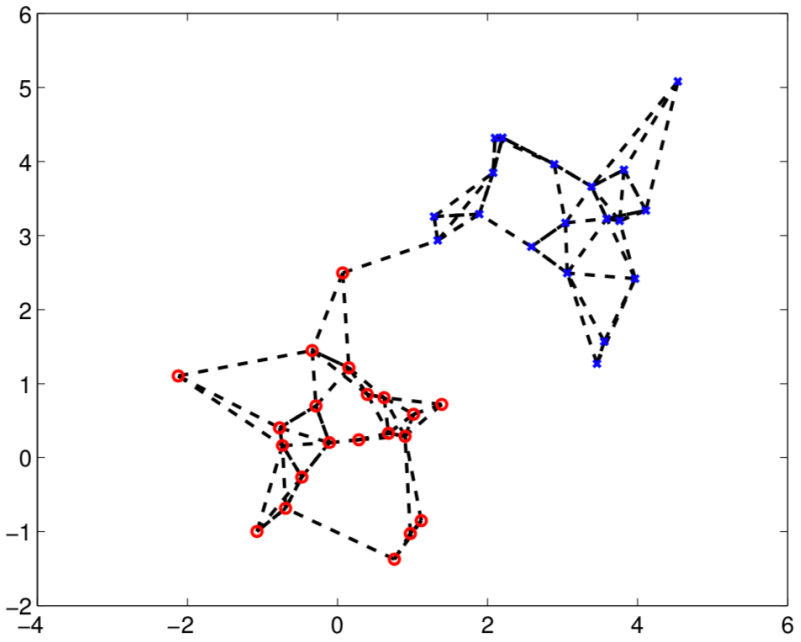
- Der generalisierte Rayleigh-Quotient ist:

$$R(A, v) = \frac{v^t A v}{v^t B v}$$

  wobei B eine symmetrische and positive-definite Matrix (alle Eigenwerte sind > 0)
- Wir verwenden für A die Laplace-Matrix L und für B die Degreematrix D

Beispiel



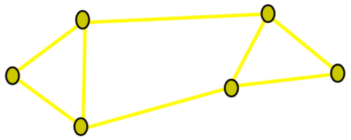


Da Genauigkeitsanforderungen bei der Clusterbildung zweitrangig, wird W dünn besetzt sein, d.h. nur wenige Eigenvektoren vorhanden. Der schnelle Algorithmus von Lanczos kann verwendet werden

Lanczos, C. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators", J. Res. Nat'l Bur. STD. 45 (4): 255–282, **1950**

Spektrale Clusterbildung

Vorverarbeitung:
Konstruiere
Laplace-Matrix L des Graphen



x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	-0.8	0	-0.2	2.5	-0.8	-0.7
x_5	-0.1	0	0	0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

Dekomposition: Finde

- Eigenwerte Λ und Eigenvektoren X von L

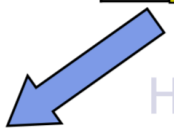
- Bilde Graphknoten auf Komponenten des zweiten Eigenvektors ab

$\Lambda =$

0.0
0.3
2.2
2.3
2.5
3.0

$X =$

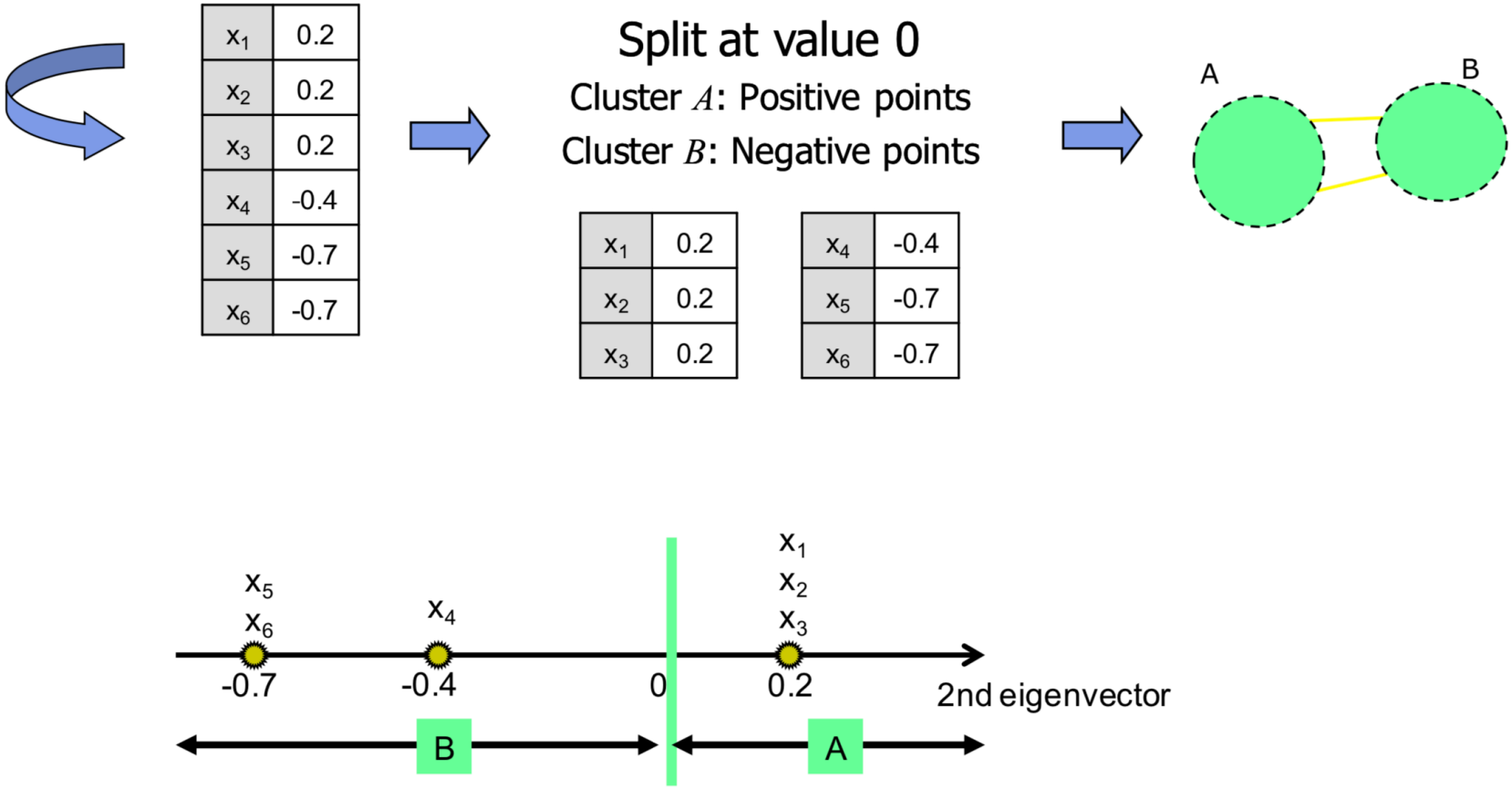
1.0	0.1	0.4	-0.2	-0.9
1.0	0.1	-0.	0.4	0.3
1.0	-0.2	0.0	-0.2	0.6
1.0	0.9	0.2	-0.4	-0.6
1.0	-0.4	-0.8	-0.6	-0.2
1.0	-0.2	0.5	0.8	0.9



x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7

How do we find the clusters?

32


x1 0.2
x2 0.2
x3 0.2
x4 -0.4
x5 -0.7
x6 -0.7
Split at value 0
Cluster A: Positive points
Cluster B: Negative points
x1 0.2
x2 0.2
x3 0.2
x4 -0.4
x5 -0.7
x6 -0.7
A
B
x5
x6
x4
x1
x2
x3
-0.7
-0.4
0
0.2
2nd eigenvector
B
A

Bi-Partition-Normalisierte Schnitte

- Bestimme gewichtete Adjazenzmatrix W (Affinitätsmatrix)
- Bestimme Diagonalmatrix D mit Einträgen $D_{ii} = \Sigma_j W_{ij}$
- Löse

$$(D - W)y = \lambda D y,$$

- Verwende Eigenvektor mit zweitkleinstem Eigenwert, um Graph in zwei Partitionen zu zerlegen
  - Zerlegung nach konstantem Wert (0 oder 0.5)
  - Zerlegung nach Median
  - Zerlegung nach kleinstem Ncut-Wert von n Zerlegungspunkten

K-Partitionierung

- Rekursive Bi-Partitionierung
  - Ineffizient, kaum stabil
- Clusterbildung mit mehreren Eigenwerten
  - Bestimme reduzierten Raum aus mehreren Eigenwerten
  - Wie Dimensionsreduktion

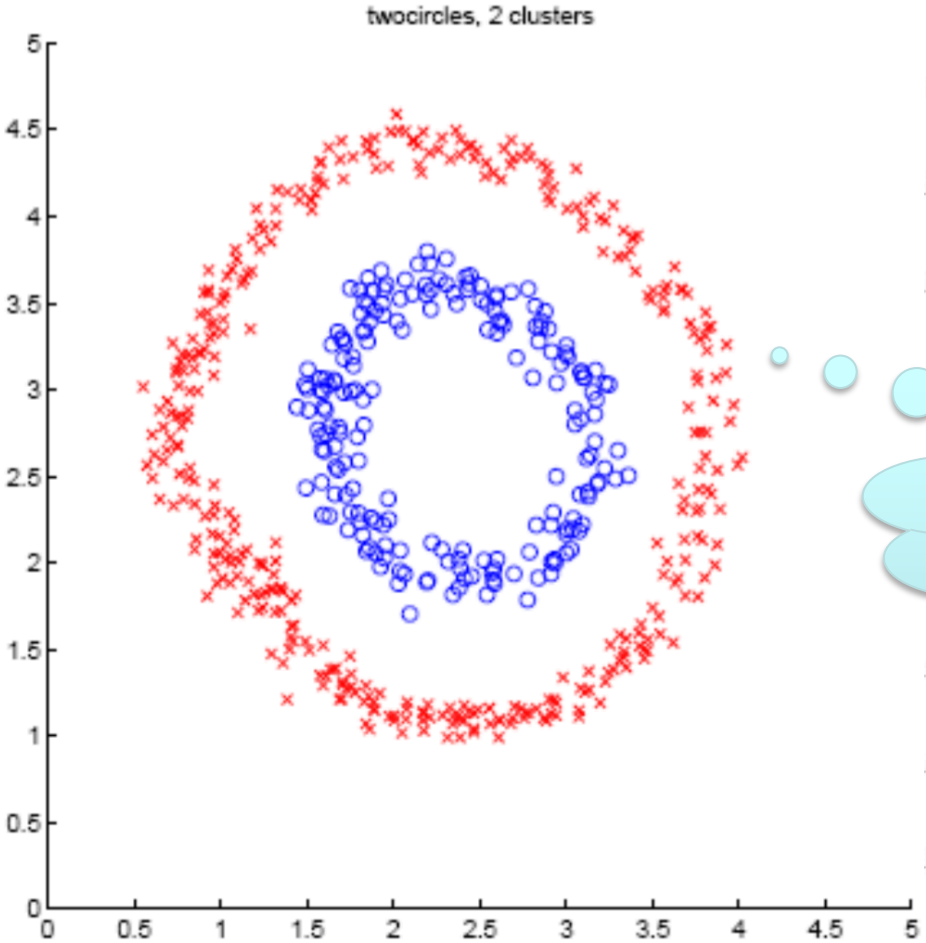
Ng, Jordan, Weiss

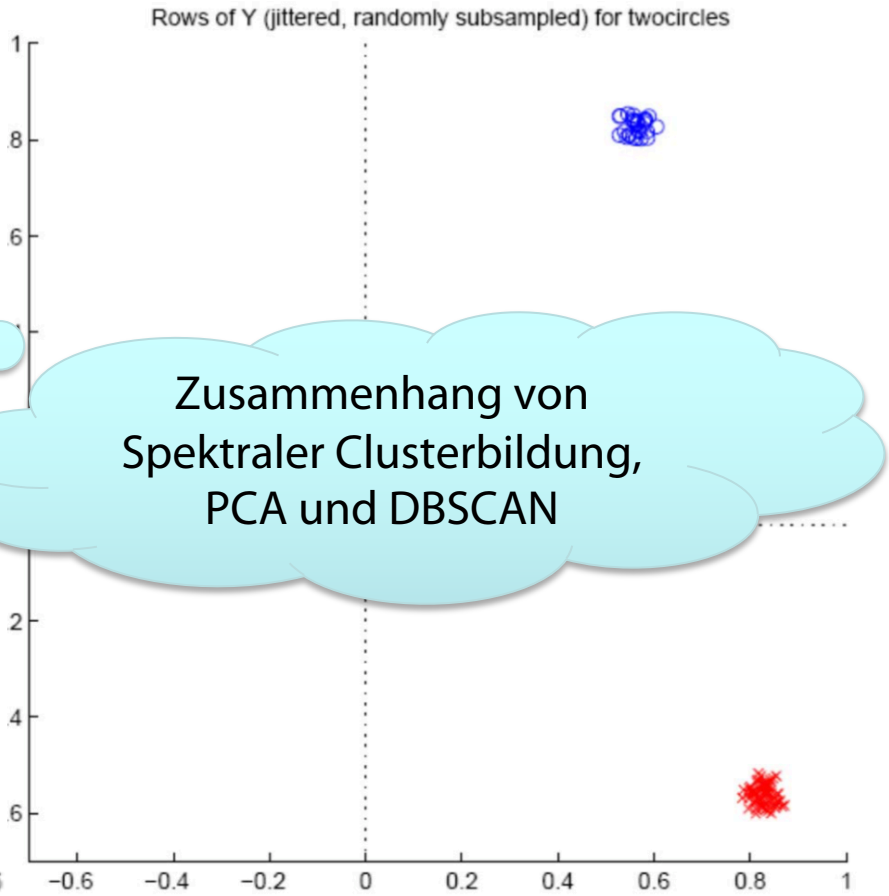
- Konstruiere Matrix $V \in \mathbb{R}^{n \times k}$
- Interpretiere Zeilen von V als Datenpunkte $Z_i \in \mathbb{R}^k$

	v_1	v_2	v_3
Z_1	v_{11}	v_{12}	v_{13}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Z_n	v_{n1}	v_{n2}	v_{n3}

- Clusterbildung der Punkte Z_i beispielsweise mit Algorithmus k-means oder ähnlich

Ng A.Y., Jordan, M.I., and Weiss Y
On Spectral Clustering: Analysis and an algorithm
In Proc. Neural Information Processing Systems **2001**



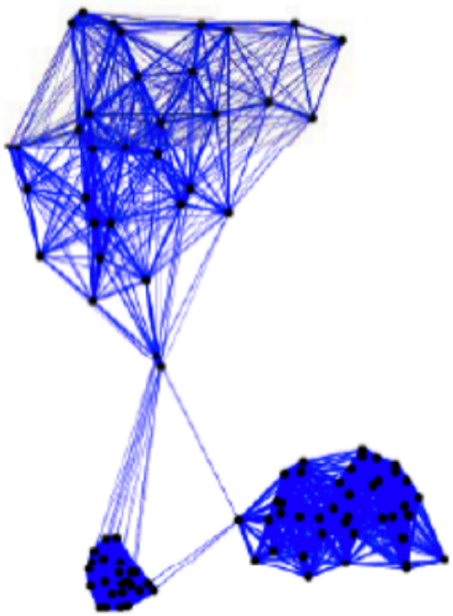


Zusammenhang von
Spektraler Clusterbildung,
PCA und DBSCAN

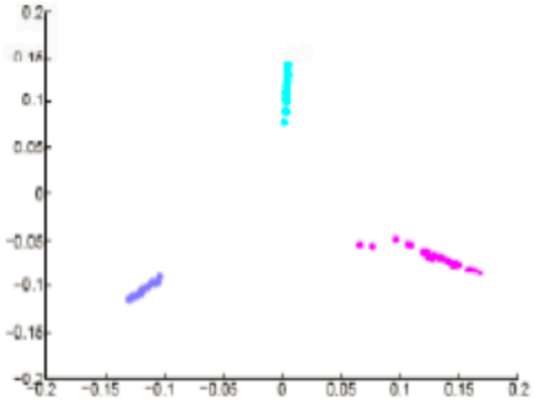
Spektrale Einbettung der Daten

Spektrale Einbettung der Daten

Graph, 20-NN



Z

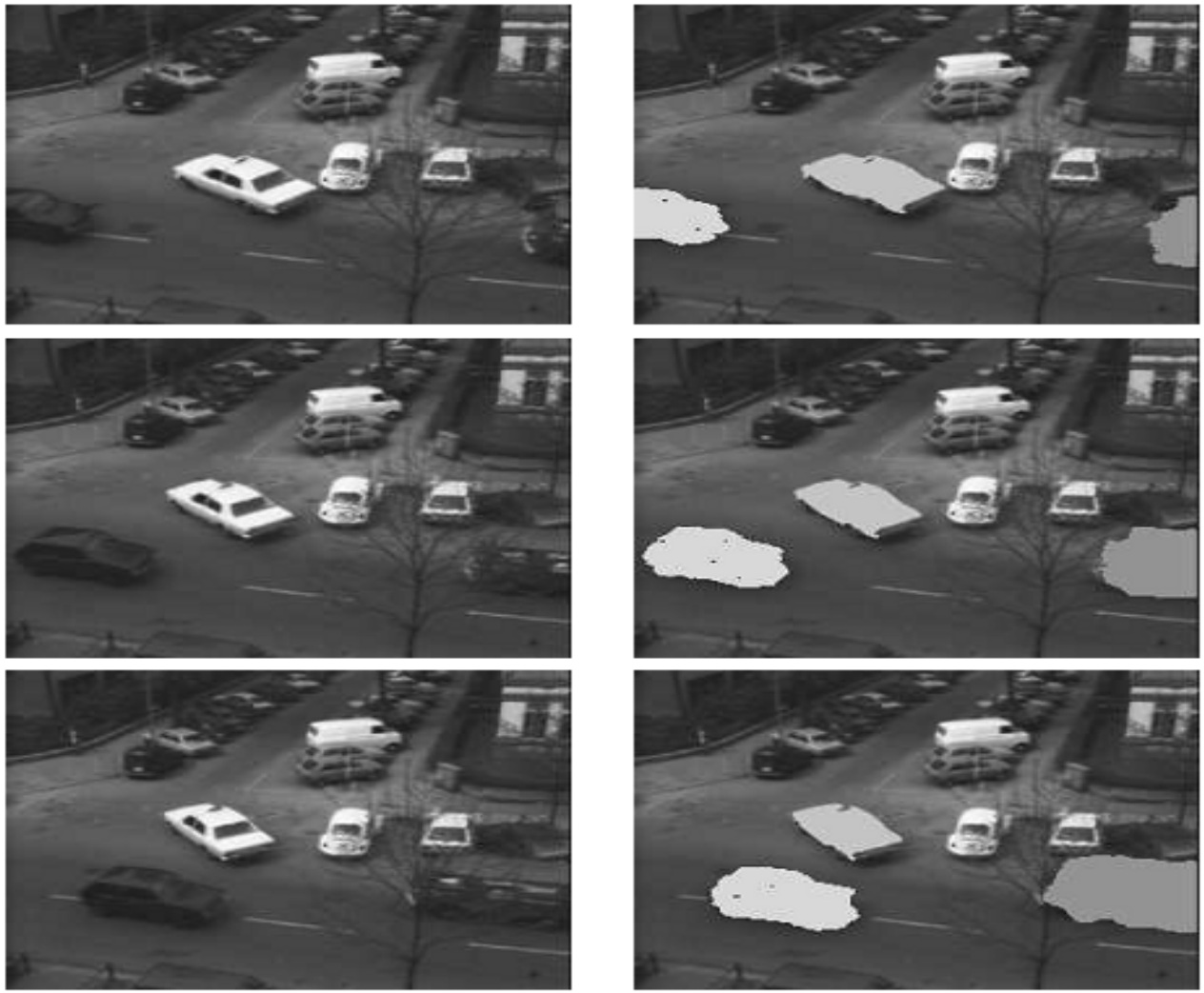


Clustering

Spektrale Clusterbildung in der Bildverarbeitung

J. Shi, J. Malik: Normalized cuts and image segmentation. In: IEEE Transactions on Pattern Analysis and Machine Intelligence. 22(8), S. 888–905, **2000**

Video-Daten = 3D-Daten



Probleme der Spektralen Clusterbildung

Ausgehend von lokaler Information aus dem gewichteten Graph, Clusterbildung global gemäß Eigenwertbestimmung

Aber:

- Normalisierter Schnitt
  kein Maß für die Qualität der Zerlegung
- Probleme mit Cluster
  verschiedener Skalierung und Dichte

Boaz Nadler and Meirav Galun. Fundamental limitations of spectral clustering. In Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS'06), 1017-1024, **2006**

Spektrale Clusterbildung

- Zeitkomplexität $O(n^3)$ (Eigenvektorberechnung)
  - Kaum für große Datenmengen geeignet
- Verbesserung unter Verwendung der Nyström-Methode
  - Approximative spektrale Clusterbildung

Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09), 907-916, **2009**

Bouneffouf, Djallel; Birol, Inanc, "Sampling with Minimum Sum of Squared Similarities for Nystrom-Based Large Scale Spectral Clustering", Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI) pp. 2313–2319, **2015**

  - Kern-basierte Verfahren: $O(d \cdot n)$ // d Merkmale

Rocco Langone, Johan A.K. Suykens, Fast kernel spectral clustering In: J. Neurocomputing, Vol: 268, Page: 27-33, **2017**

Hierarchische Clusterbildung: Zusammenfassung

- CURE: empirisch $O(n^2)$ (1998)
  - Heuristische Approximation der spektralen Clusterbildung
  - Fokus auf: Wie rechnet man etwas aus
- BIRCH: empirisch $O(n \cdot \log n)$ (1996)
  - Stärkere heuristische Approximation der spektralen Clusterbildung
  - Fokus auf: Wie rechnet man etwas aus
- Spektrale Clusterbildung: $O(n^3)$ bis zu $O(d \cdot n)$ (2001-heute)
  - Schöne mathematische Beschreibung des Clusterbildungsproblems
  - Richtiger Ansatz mit Fokus auf: Was soll ausgerechnet werden?
  - Wie weit entfernt ist man bei Approximationen noch von BIRCH?
- …

Nichts geht ohne multidimensionale Indexstrukturen