Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

Skript zur Vorlesung

Knowledge Discovery in Databases

im Sommersemester 2011

Kapitel 5: Clustering

Vorlesung+Übungen:
PD Dr. Matthias Schubert, Dr. Arthur Zimek

http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_I_(KDD_I)

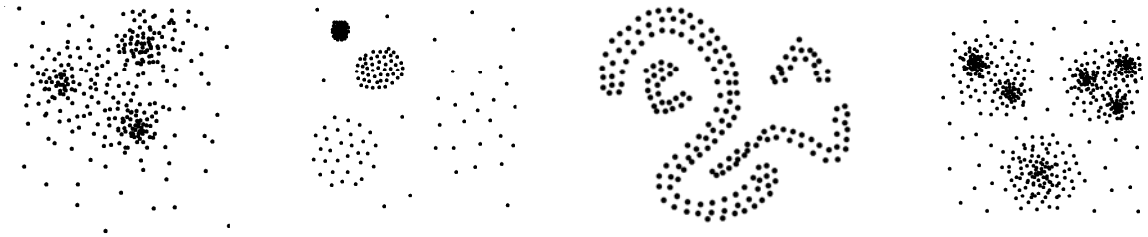
5.1 Einleitung

Ziel des Clustering

Identifikation einer endlichen Menge von Kategorien, Klassen oder Gruppen (*Cluster*) in den Daten.

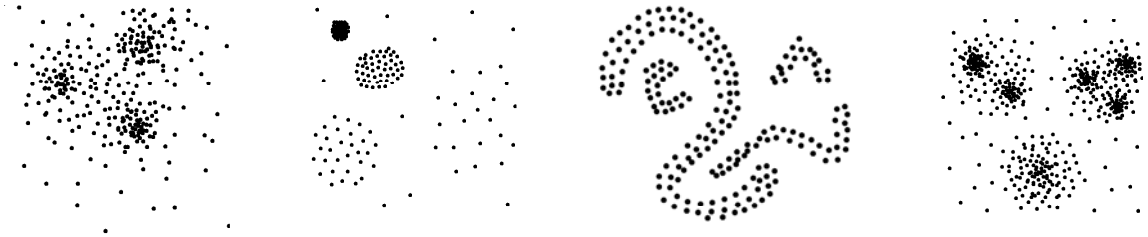
Ähnliche Objekte sollen im *gleichen* Cluster sein, *unähnliche* Objekte sollen in *unterschiedlichen* Clustern sein.

Clustering ist im Prinzip eine Art Klassifikation, allerdings ohne Trainingsdaten



Unterschied zur Klassifikation

- Clustering ist unsupervised, d.h.
  - keine Regeln zur Einordnung der Punkte in Clustern lernbar
  - wir wissen nicht wie viele Cluster vorhanden sind
  - wir wissen nicht, wie die einzelnen Cluster charakterisiert sind



- Da wir keine Trainingsdaten haben, brauchen wir andere Methoden um die Güte eines Clusterings zu beurteilen

- Herausforderungen
  - Gegeben eine Funktion *f*, die für eine Menge von Datenobjekten entscheidet, ob sie einen Cluster bilden
  - Die naive Methode wäre, für alle möglichen Teilmengen an Objekten die Funktion *f* auswerten
  - Problem:
    - Es gibt $O(2^n)$ viele Teilmengen
    - Wie sieht diese Funktion *f* überhaut aus?
  - Lösung: wir brauchen ***Heuristiken*** für beide Teilprobleme
    - Effiziente Suche nach Lösungen
    - Effiziente und effektive Modellierung von *f*

  => es gibt sehr viele verschiedene Clusteringalgorithmen

Typen von Clustering Verfahren

- Partitionierende Verfahren
  - Modell: Cluster sind kompakt zusammenliegende Datenobjekte
  - Parameter: (meist) Anzahl *k* der Cluster (d.h. Annahme: Anzahl der Cluster bekannt), Distanzfunktion
  - sucht ein „flaches" Clustering in *k* Cluster mit maximaler Kompaktheit
- Dichte-basierte Verfahren
  - Modell: Cluster sind Räume mit hoher Punktdichte separiert durch Räume niedriger Punktdichte
  - Parameter: minimale Dichte in einem Cluster, Distanzfunktion
  - sucht flaches Clustering: erweitert Punkte um ihre Nachbarn solange Dichte groß genug

Typen von Clustering Verfahren

- Hierarchische Verfahren
  - Modell: Kompaktheit, Dichte, …
  - Parameter: Distanzfunktion für Punkte und für Cluster
  - bestimmt Hierarchie von Clustern (z.B. in Form eines Baumes darstellbar), mischt jeweils die ähnlichsten Cluster
  - Flaches Clustering kann durch Abschneiden des Baumes erzeugt werden
- Andere Clustering-Verfahren
  - Fuzzy Clustering
  - Graph-theoretische Verfahren
  - Neuronale Netze
  - …

Grundlagen

Ziel: Partitionierung in *k* Cluster, so dass eine Kostenfunktion minimiert wird (Gütekriterium: Kompaktheit)

Zentrale Annahmen:

- Anzahl *k* der Cluster bekannt (Eingabeparameter)
- Clustercharakteristik: Kompaktheit
- Kompaktheit: Abweichung aller Objekte im Cluster von einem ausgezeichneten Cluster-Repräsentanten ist minimal
- Kompaktheitskriterium führt meistens zu sphärisch geformten Clustern

DATABASE SYSTEMS GROUP
5.2 Partitionierende Verfahren
LMU

Grundlagen

Erschöpfende Suche ist zu ineffizient (WARUM?)

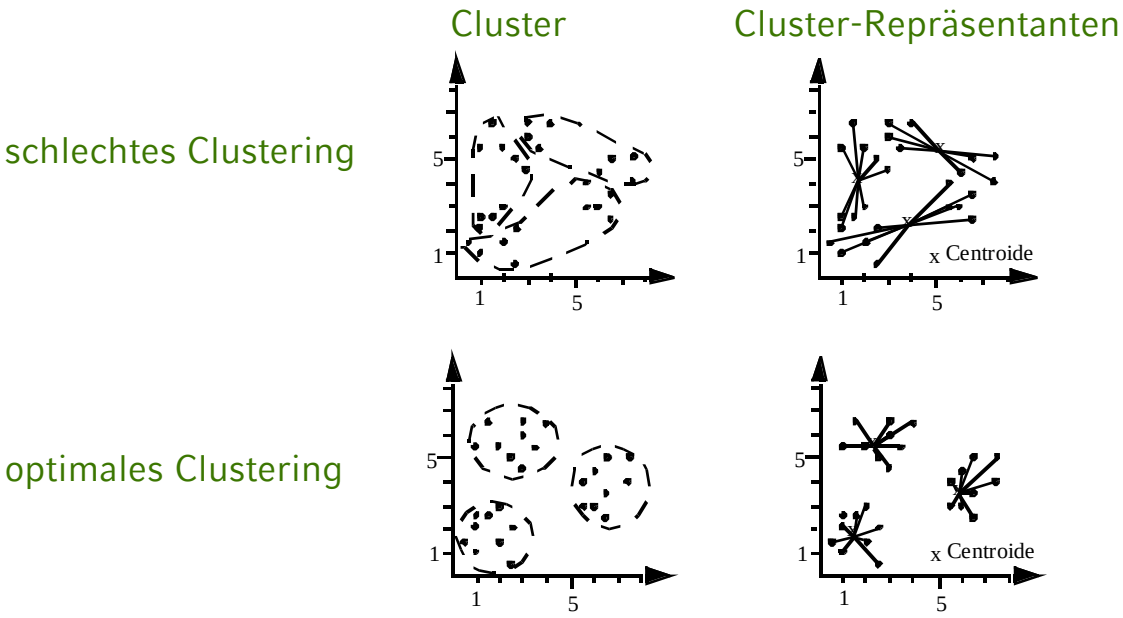
Daher: Lokal optimierende Verfahren

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

Typen von Cluster-Repräsentanten

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

Konstruktion zentraler Punkte (Beispiel)



Konstruktion zentraler Punkte (Grundbegriffe)

[Forgy 1965]

- Objekte sind Punkte $x = (x_1, ..., x_d)$ in einem euklidischen Vektorraum

  dist = euklidische Distanz (L_2-Norm)

- *Centroid* μ_C: Mittelwert aller Punkte im Cluster *C*

- *Maß für die Kosten* (Kompaktheit) *eines Clusters C*

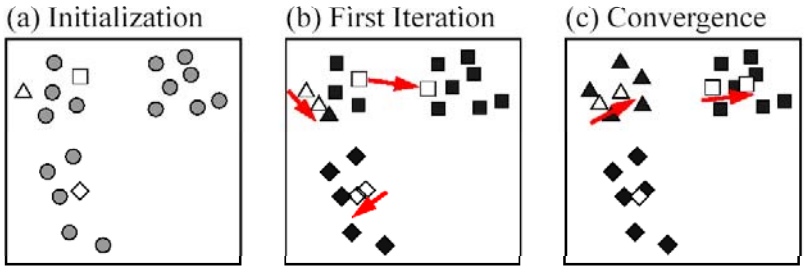
$$TD^2(C) = \sum_{p \in C} dist(p, \mu_C)^2$$

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

$$TD^2(C_1, \ldots, C_k) = \sum_{i=1}^{k} TD^2(C_i)$$

Idee des Algorithmus

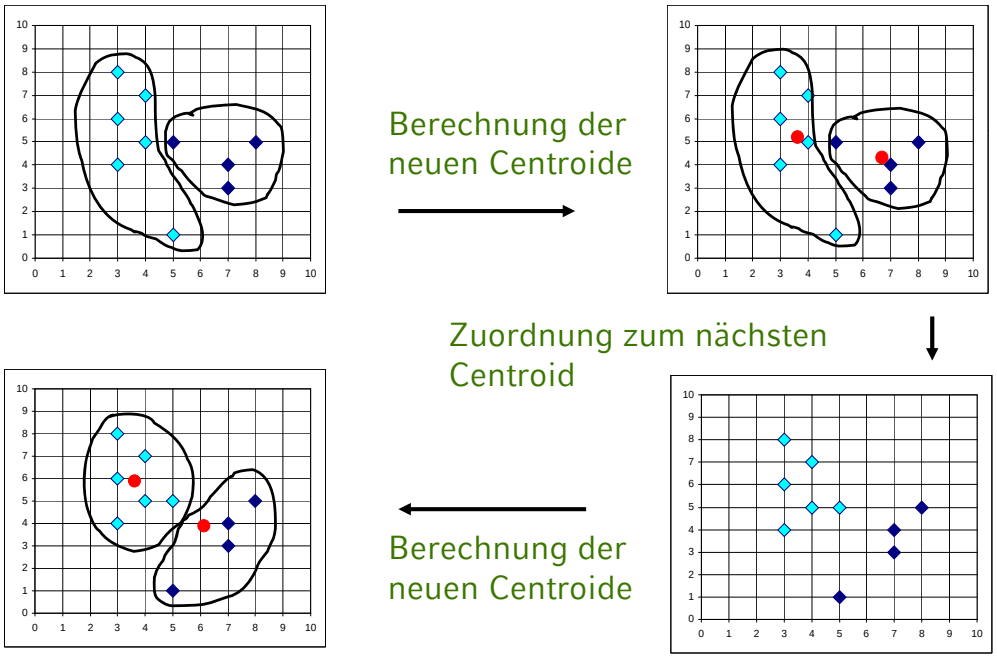
- Algorithmus startet z.B. mit zufällig gewählten Punkten als Cluster-Repräsentanten
- Der Algorithmus besteht aus zwei alternierenden Schritten:
  - Zuordnung jedes Datenpunktes zum räumlich nächsten Repräsentanten
  - Neuberechnung der Repräsentanten (Centroid der zugeordneten Punkte)
- Diese Schritte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



Algorithmus

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

Beispiel



Varianten des Basis-Algorithmus

k-means [MacQueen 67]

Idee: die betroffenen Centroide werden direkt aktualisiert, wenn ein Punkt seine Clusterzugehörigkeit ändert

- *k*-means hat im wesentlichen die Eigenschaften des Basis-Algorithmus
- *k*-means ist aber reihenfolgeabhängig

ISODATA

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

Diskussion

+ Effizienz
  Aufwand: O($k \cdot n$) für eine Iteration (k kann für kleine Anzahl von Clustern vernachlässigt werden), Anzahl der Iterationen ist im allgemeinen klein (~ 5 - 10).
+ einfache Implementierung

  $\Rightarrow$ *k*-means ist das populärste partitionierende Clustering-Verfahren

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

Auswahl repräsentativer Punkte

[Kaufman & Rousseeuw 1990]

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

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

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

$$TD(C_1, \ldots, C_k) = \sum_{i=1}^{k} TD(C_i)$$

Überblick über k-medoid Algorithmen

PAM [Kaufman & Rousseeuw 1990]

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

CLARANS [Ng & Han 1994]

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

Algorithmus PAM

```
PAM(Punktmenge D, Integer k)
   Initialisiere die k Medoide;
   TD_Änderung := −∞;
   while TD_Änderung < 0 do
      Berechne für jedes Paar (Medoid M, Nicht-Medoid
        N) den Wert TD_{N↔M};
      Wähle das Paar (M, N), für das der Wert
        TD_Änderung := TD_{N↔M} − TD minimal ist;
      if TD_Änderung < 0 then
         ersetze den Medoid M durch den Nicht-Medoid N;
         Speichere die aktuellen Medoide als die bisher beste
           Partitionierung;
   return Medoide;
```

Algorithmus CLARANS

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


Vergleich von PAM und CLARANS

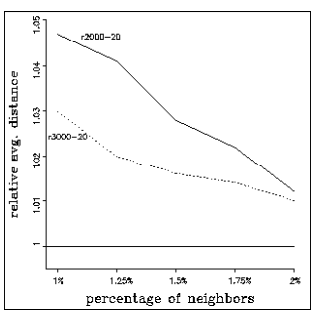
Laufzeitkomplexitäten

PAM: $O(n^3 + k(n-k)^2 * \#Iterationen)$

CLARANS: $O(numlocal * maxneighbor * \#Ersetzungen * n)$
praktisch $O(n^2)$

Experimentelle Untersuchung

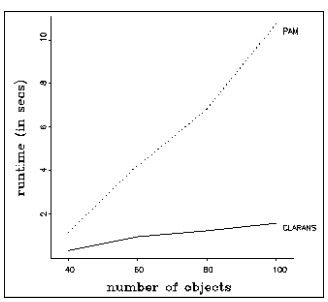
Qualität



TD(CLARANS)

TD(PAM)

Laufzeit



Erwartungsmaximierung (EM)

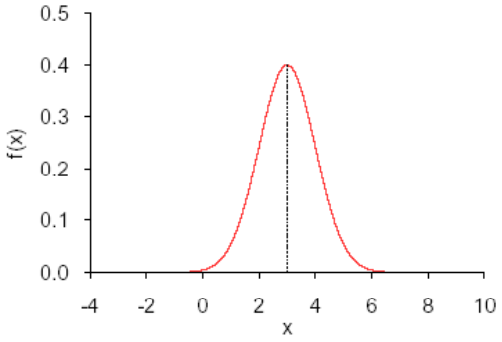
[Dempster, Laird & Rubin 1977]

- Objekte sind Punkte x = (x_1, ..., x_d) in einem euklidischen Vektorraum
- Ein Cluster wird durch eine Wahrscheinlichkeitsverteilung beschrieben
- typisch: Modell für einen Cluster ist eine multivariate Normalverteilung
- Repräsentation eines Clusters *C*
  - Mittelwert μ_C aller Punkte des Clusters (Centroid)
  - *d x d* Kovarianzmatrix Σ_C für die Punkte im Cluster *C*
- Wahrscheinlichkeitsdichte eines Clusters *C*

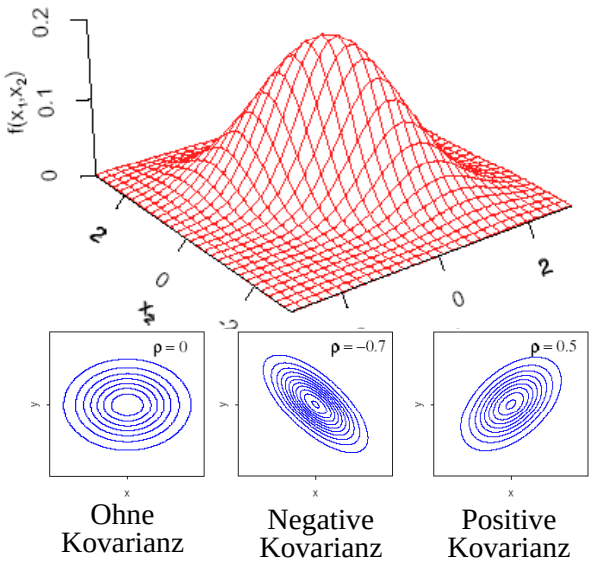
$$P(x \mid C) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_C \mid}} e^{-\frac{1}{2}\cdot(x-\mu_C)^T \cdot (\Sigma_C)^{-1} \cdot (x-\mu_C)}$$

Multivariate Normalverteilung

Univariate Normalverteilung



Bivariate Normalverteilung



5.2 Partitionierende Verfahren

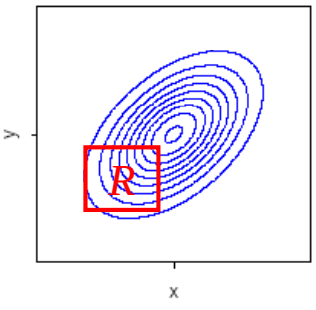
Idee des EM-Algorithmus:

- Jeder Punkt gehört zu mehreren (eigentlich *allen*) Clustern, jeweils mit unterschiedlicher Wahrscheinlichkeit, abh. v. $P(x|C)$
- Algorithmus besteht wieder aus zwei alternierenden Schritten:
  - Zuordnung von Punkten zu Clustern (hier nicht absolut sondern relativ/nach Wahrscheinlichkeit)
  - Neuberechnung der Cluster-Repräsentanten (Gauß-Kurven)
- Alles muss auf eine stochastische Grundlage gestellt werden:
  - Bei Berechnung der Clusterzentren (μ_C) muss berücksichtigt werden, dass Punkte Clustern nicht absolut sondern nur relativ zugeordnet sind
  - Wie groß ist die Wahrscheinlichkeit der Clusterzugehörigkeit?

5.2 Partitionierende Verfahren

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

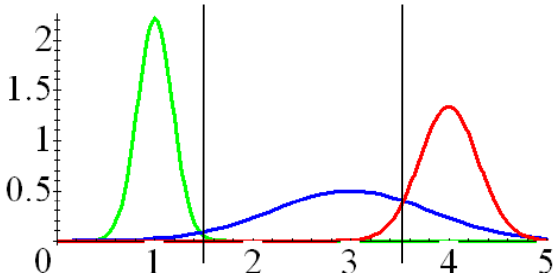
$$P(x \mid C_i) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_{C_i} \mid}} e^{-\frac{1}{2}\cdot(x-\mu_{C_i})^T\cdot(\Sigma_{C_i})^{-1}\cdot(x-\mu_{C_i})}$$



Dichtefunktion:

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

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



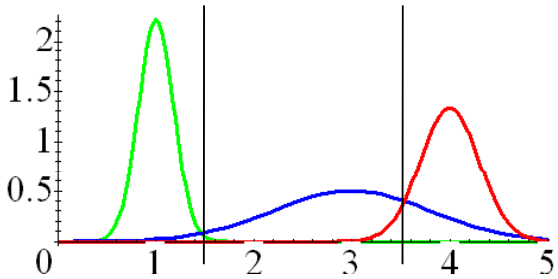
Bedingte Wahrscheinlichkeit:

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

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

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

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



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

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

5.2 Partitionierende Verfahren

- Maß für die Güte eines Clustering *M*

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

⇨ *E*(*M*) soll maximiert werden.

- Anteil des Clusters an der Datenmenge:

$$W_i = P(C_i) = \frac{1}{n} \sum_{j=1}^{n} P(C_i \mid x_j)$$

- Mittelwert und Kovarianzmatrix der Gaußverteilung:

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

5.2 Partitionierende Verfahren

Algorithmus

```
ClusteringDurchErwartungsmaximierung(Punktmenge D, Integer k)
   Erzeuge ein „initiales“ Modell M’ = (C1’, ..., Ck’);
   repeat // „Neuzuordnung“
      Berechne P(x|Ci), P(x) und P(Ci|x) für jedes Objekt aus
        D und jede Gaußverteilung/jeden Cluster Ci;
      // „Neuberechnung des Modells“
      Berechne ein neues Modell M ={C1, ..., Ck} durch
        Neuberechnung von Wi, μC und ΣC für jedes i;
      M’ := M;
   until |E(M) - E(M’)| < ε;
   return M;
```

Diskussion

- Aufwand:

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

  Anzahl der benötigten Iterationen im allgemeinen sehr hoch

- Ergebnis und Laufzeit hängen (wie beim *k-means* und *k-medoid*) stark ab
  - von der initialen Zuordnung
  - von der „richtigen" Wahl des Parameters *k*
- Modifikation für Partitionierung der Daten in *k disjunkte* Cluster:
  jedes Objekt nur demjenigen Cluster zuordnen, zu dem es am wahrscheinlichsten gehört.

Wahl des initialen Clusterings

Idee

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

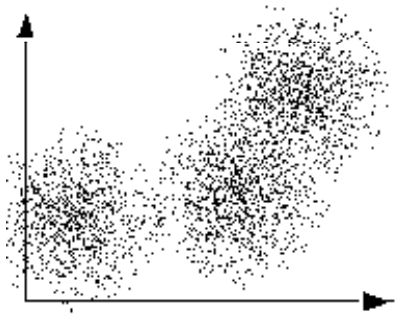
Methode [Fayyad, Reina & Bradley 1998]

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

  ⇨ *m* verschiedene Schätzungen für *k* Clusterzentren

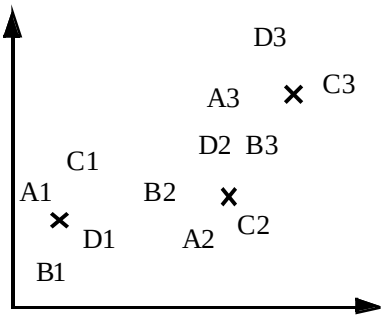
  $A = (A_1, A_2, \ldots, A_k)$, $B = (B_1, \ldots, B_k)$, $C = (C_1, \ldots, C_k)$, . . .
- Clustere nun die Menge $DB = A \cup B \cup C$..
  mit *m* verschiedenen Stichproben *A*, *B*, *C*, . . . als Startkonfiguration
- Wähle von den *m* Clusterings dasjenige mit dem besten Wert bezüglich des zugehörigen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

Beispiel



Grundgesamtheit

$k = 3$ Gauß-Cluster



Clusterzentren

von $m = 4$ Stichproben

× *wahre Clusterzentren*

Wahl des Parameters k

Methode

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

Maß für die Güte eines Clusterings

- muss unabhängig von der Anzahl k sein
- bei k-means und k-medoid: TD^2 und TD sinken monoton mit steigendem k
- bei EM: E steigt monoton mit steigendem k
- Brauche ein von k unabhängiges Gütemaß für die k-means- und k-medoid-Verfahren

⇨ *Silhouetten-Koeffizient*

Silhouetten-Koeffizient [Kaufman & Rousseeuw 1990]

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

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

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

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

- Silhouettenkoeffizient s_C eines Clustering durchschnittliche Silhouette aller Objekte
- Interpretation des Silhouettenkoeffizients

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

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

k-modes Verfahren [Huang 1997]

- *k*-medoid-Algorithmus wesentlich langsamer als *k*-means- Algorithmus
- *k*-means-Verfahren nicht direkt für kategorische Attribute anwendbar

=> gesucht ist ein Analogon zum Centroid eines Clusters

- Numerische Attribute
  Centroid $\bar{x}$ einer Menge C von Objekten minimiert $TD(C, \bar{x}) = \sum_{p \in C} dist(p, \bar{x})$
- Kategorische Attribute
  Mode m einer Menge C von Objekten minimiert $TD(C, m) = \sum_{p \in C} dist(p, m)$
  (m ist nicht unbedingt ein Element der Menge C)
- $m = (m_1, ..., m_d)$, *dist* eine Distanzfunktion für kategorische Attribute, z.B.

$$dist(x, y) = \sum_{i=1}^{d} \delta(x_i, y_i) \text{ mit } \delta(x_i, y_i) = \begin{cases} 0, & falls\ x_i = y_i \\ 1, & sonst \end{cases}$$

Bestimmung des Modes

- Die Funktion $TD(C,m) = \sum_{p \in C} dist(p,m)$ wird genau dann minimiert, wenn für $m = (m_1, ..., m_d)$ und für alle Attribute A_i, $i = 1,..., d$, gilt:

  Es gibt in A_i keinen häufigeren Attributwert als m_i

- Der Mode einer Menge von Objekten ist nicht eindeutig bestimmt.

- Beispiel

  Objektmenge {(a, b), (a,c), (c, b), (b,c)}
  (a, b) ist ein Mode
  (a, c) ist ein Mode

Algorithmus k-modes

- Initialisierung

  keine zufällige Partitionierung
  sondern *k* Objekte aus der Datenmenge als initiale *Modes*

- Cluster-Repräsentanten

  Mode anstelle des Centroids

- Distanzfunktion

  anstelle der quadrierten euklidischen Distanz

  Distanzfunktion für Datensätze mit kategorischen Attributen

5.3 Dichtebasiertes Clustering

Grundlagen

Idee

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

Zentrale Annahmen

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

DATABASE SYSTEMS GROUP

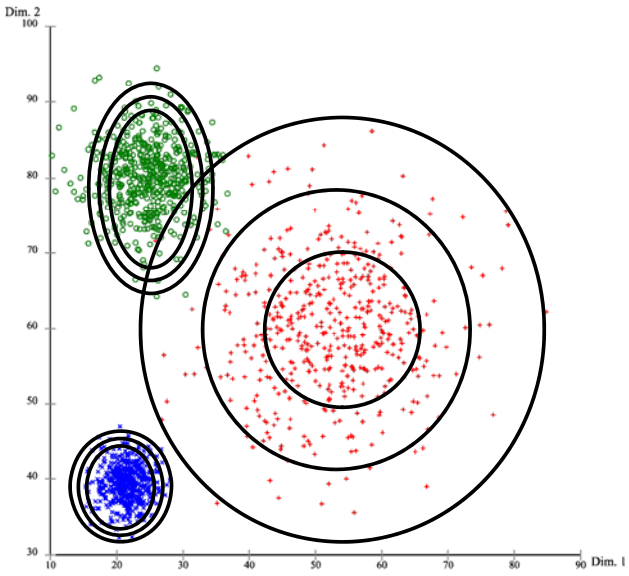
5.3 Dichtebasiertes Clustering

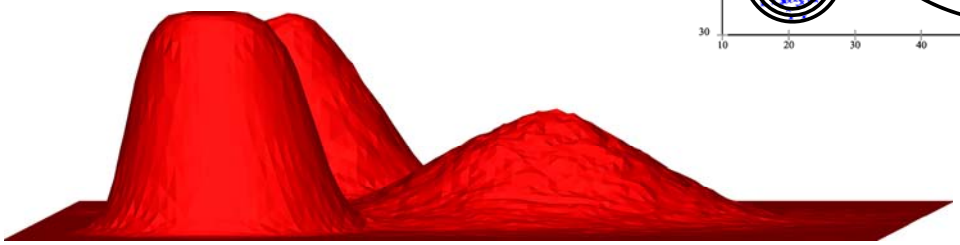
LMU

Intuition

(nach: [Kriegel, Kröger, Sander, Zimek 2011])

- Partitionierend: approximiere unterschiedliche Dichtefunktionen
- Dichte-basiert: unabhängig von zugrundeliegender Dichtefunktion: selektiere Bereiche, die mit Mindest-Dichte verbunden sind

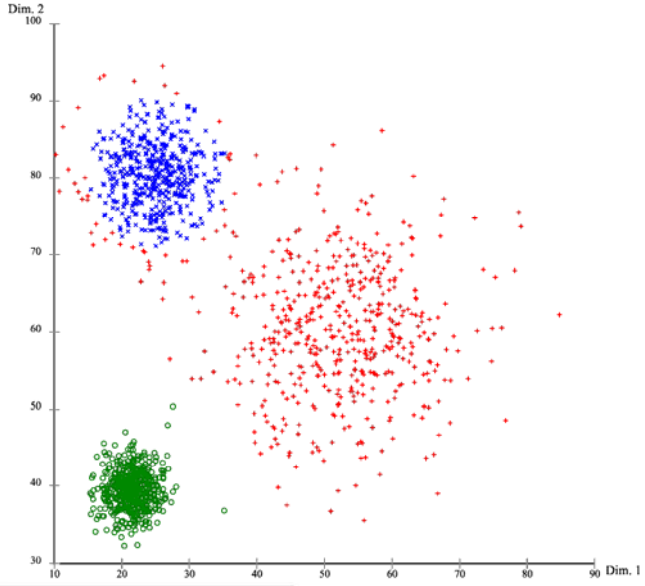




5.3 Dichtebasiertes Clustering

Intuition

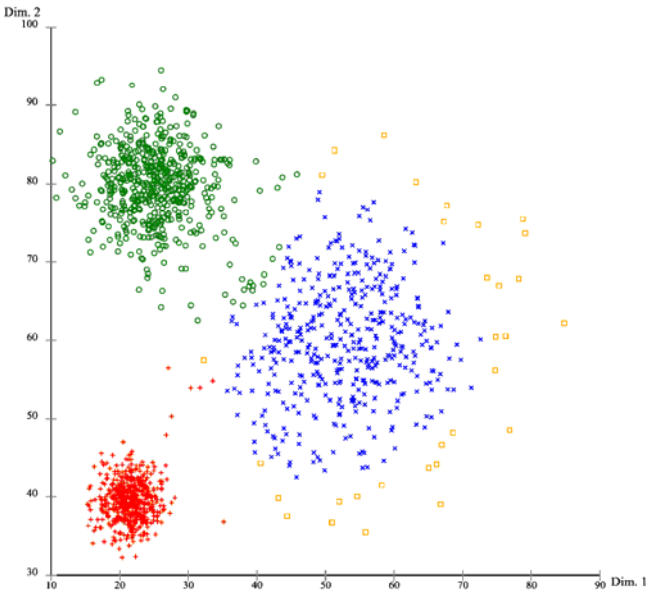
hohes Dichte-Niveau: Cluster
geringer Dichte gilt als Noise

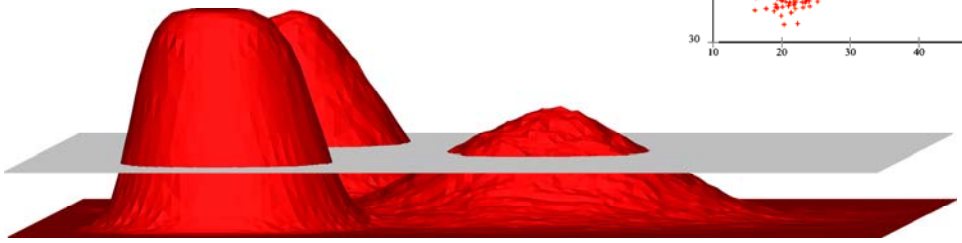


5.3 Dichtebasiertes Clustering

Intuition

mittleres Dichte-Niveau:
Identifikation von drei Clustern,
einige Punkte in den Ausläufern
der Verteilungen werden zu
Noise

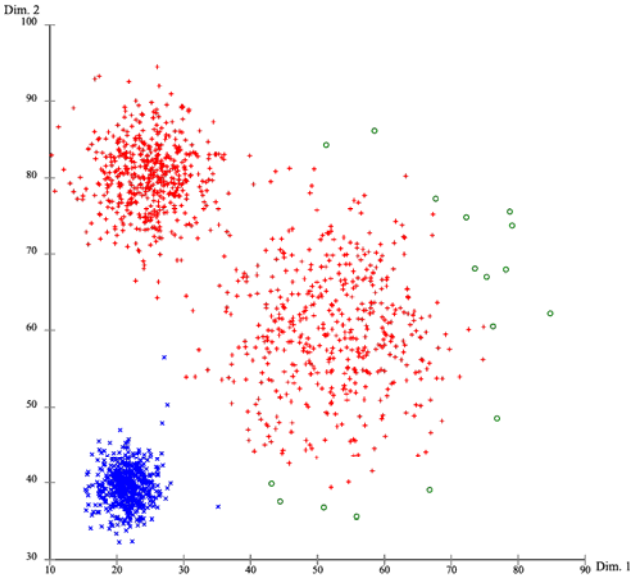




5.3 Dichtebasiertes Clustering

Intuition

geringes Dichte-Niveau: zwei Cluster verschmelzen



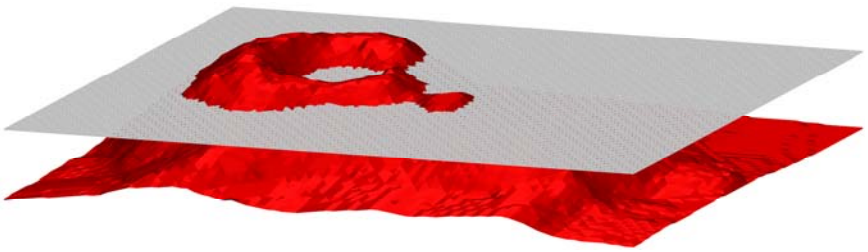
5.3 Dichtebasiertes Clustering

Intuition

Partitionierend: Optimierungsproblem

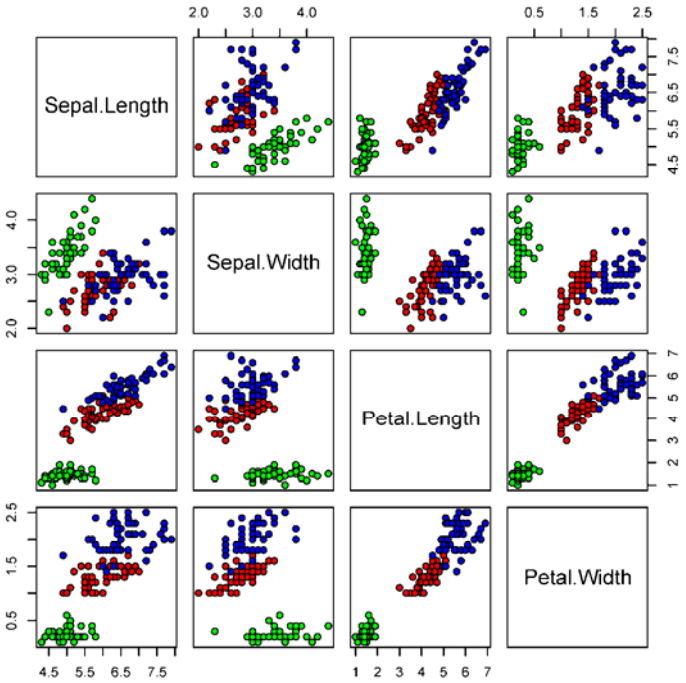
Dichte-basiert: „natürliche“ Strukturen (oft: geographisch/ topographisch, oder biologisch)

Beispiel: Höhenprofil (Vulkan Mt. Eden, Auckland)



Intuition

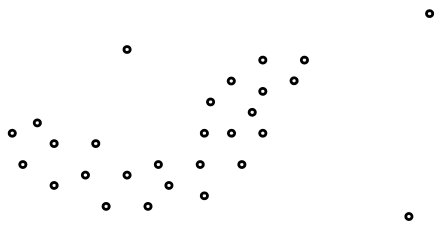
Beispiel: Iris-Arten – innerhalb einer Art gewisse Bandbreite, aber keine Sprünge



Idee zur effizienten Suche

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

ε $MinPts = 4$

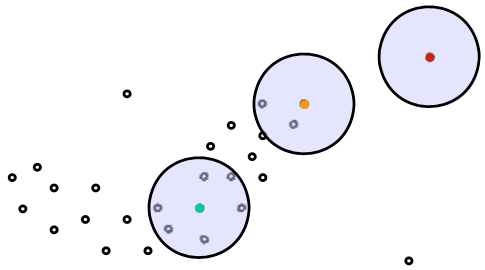


Idee zur effizienten Suche

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

ε $MinPts = 4$

- *Kernpunkt*

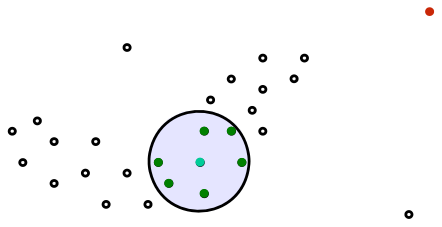


Idee zur effizienten Suche

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

ε $MinPts = 4$

- *Kernpunkt*
- *Direkte Dichte-Erreichbarkeit*

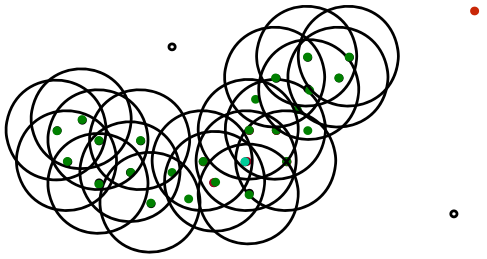


Idee zur effizienten Suche

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

ε $MinPts = 4$

- *Kernpunkt*
- *Direkte Dichte-Erreichbarkeit*
- *Dichte-Erreichbarkeit*



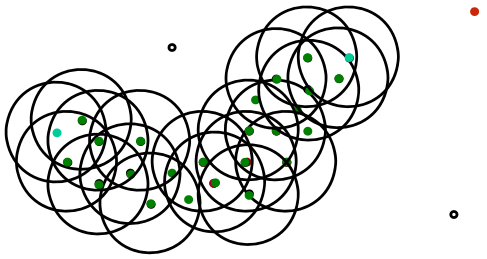
DATABASE SYSTEMS GROUP
5.3 Dichtebasiertes Clustering
LMU

Idee zur effizienten Suche

Parameter $\varepsilon \in \Re$ und $MinPts \in \aleph$ spezifizieren Dichtegrenzwert

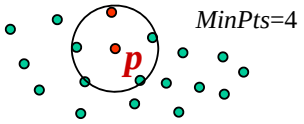
ε $MinPts = 4$

- *Kernpunkt*
- *Direkte Dichte-Erreichbarkeit*
- *Dichte-Erreichbarkeit*
- *Dichte-Verbundenheit*

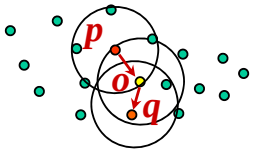


Formalisierung [Ester, Kriegel, Sander & Xu 1996]

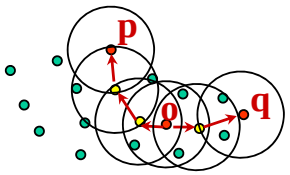
- Ein Objekt $p \in DB$ heißt *Kernobjekt*, wenn gilt:
  $|RQ(p,\varepsilon)| \geq MinPts$
  $RQ(p,\varepsilon) = \{o \in DB \mid dist(p,o) <= \varepsilon\}$



- Ein Objekt $p \in DB$ ist *direkt dichte-erreichbar* von $q \in DB$ bzgl. ε und *MinPts*, wenn gilt:
  $p \in RQ(q,\varepsilon)$ und q ist ein Kernobjekt in *DB*.



- Ein Objekt p ist *dichte-erreichbar* von q, wenn es eine Kette von direkt erreichbaren Objekten von q nach p gibt.
- Zwei Objekte p und q sind *dichte-verbunden*, wenn sie beide von einem dritten Objekt o aus dichte-erreichbar sind.



Formalisierung

Ein (*dichtebasierter*) *Cluster* C bzgl. ε und *MinPts* ist eine nicht-leere Teilmenge von *DB* für die die folgenden Bedingungen erfüllt sind:

Maximalität: $\forall p,q \in DB$: 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.

Formalisierung

Definition Clustering

Ein *dichtebasiertes Clustering* *CL* der Menge *DB* bzgl. ε und *MinPts* ist eine „vollständige" Menge von dichtebasierten Clustern bzgl. ε und *MinPts* in *DB*.

Definition Noise

Die Menge $Noise_{CL}$ („*Rauschen*") ist definiert als die Menge aller Objekte aus *DB*, die nicht zu einem der dichtebasierten Cluster $C \in CL$ gehören.

Grundlegende Eigenschaft

Sei *C* ein dichtebasierter Cluster und sei $p \in C$ ein Kernobjekt.
Dann gilt:

$$C = \{o \in DB \mid o \text{ dichte-erreichbar von } p \text{ bzgl. } \varepsilon \text{ und } MinPts\}.$$

=> ermöglicht effiziente Suche (WARUM?)

Algorithmus DBSCAN

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

   ClusterId := nextId(NOISE);
   for i from 1 to |DB| do
      Objekt := DB.get(i);
      if Objekt.ClId = UNKLASSIFIZIERT then
         if ExpandiereCluster(DB, Objekt, ClusterId, ε, MinPts)
         then ClusterId:=nextId(ClusterId);
```

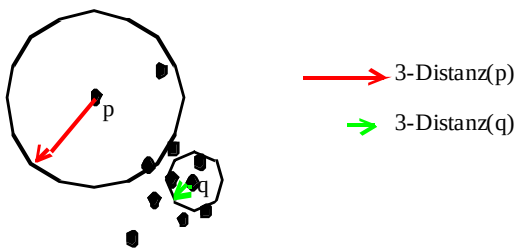
```
ExpandiereCluster(DB, StartObjekt, ClusterId, ε, MinPts): Boolean
seeds:= RQ(StartObjekt, ε);
if |seeds| < MinPts then // StartObjekt ist kein Kernobjekt
  StartObjekt.ClId := NOISE;
  return false;
// sonst: StartObjekt ist ein Kernobjekt
forall o ∈ seeds do o.ClId := ClusterId;
  entferne StartObjekt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


Parameterbestimmung

Cluster: Dichte größer als die durch ε und *MinPts* spezifizierte „Grenzdichte“

Gesucht: der am wenigsten dichte Cluster in der Datenmenge

Heuristische Methode: betrachte die Distanzen zum *k*-nächsten Nachbarn.

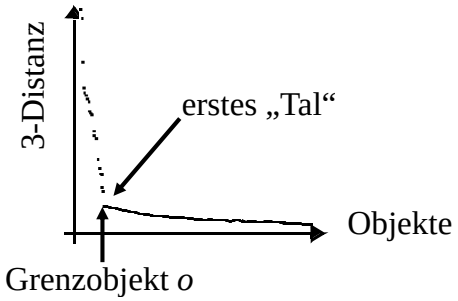


Funktion *k-Distanz*: Distanz eines Objekts zu seinem *k*-nächsten Nachbarn

k-Distanz-Diagramm: die *k*-Distanzen aller Objekte absteigend sortiert

Parameterbestimmung

Beispiel eines *k*-Distanz-Diagramms

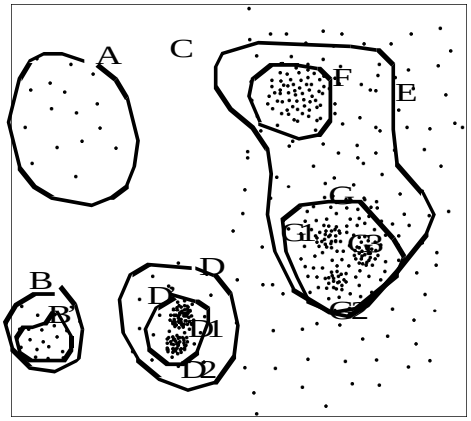


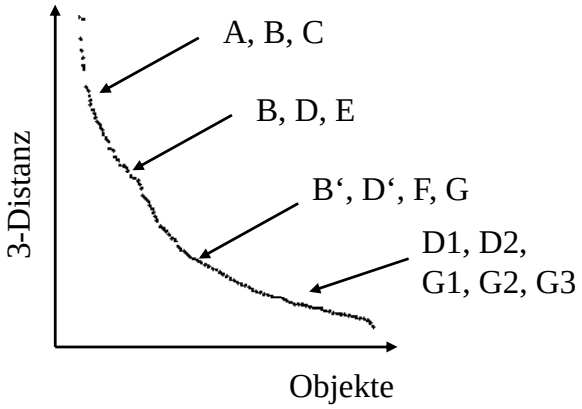
Heuristische Methode

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

Probleme der Parameterbestimmung

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

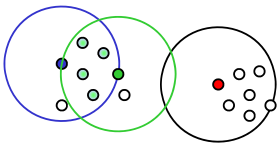




Shared Nearest Neighbor (SNN) Clustering

- DBSCAN
  - Erkennt Cluster unterschiedlicher Form und Größe
  - Hat Probleme bei Clustern mit unterschiedlicher Dichte
- Verbesserung: anderer Ähnlichkeitsbegriff
  - Ähnlichkeit zwischen zwei Objekten, wenn sie beide sehr nahe zu einer Referenzmenge *R* sind
  - Ähnlichkeit wird durch die Referenzmenge *R* "bestätigt"
  - Ähnlichkeit z.B. durch die Anzahl gemeinsamer nächster Nachbarn definieren (d.h. *R* ist die Menge der nächsten Nachbarn)
  - Shared Nearest Neighbor (SNN) Ähnlichkeit:
    - SNN_k-similarity(p,q) = $|NN(p, k) \cap NN(q, k)|$
    - $NN(o, k)$ = Menge der k-nächsten Nachbarn von o

$k = 6$

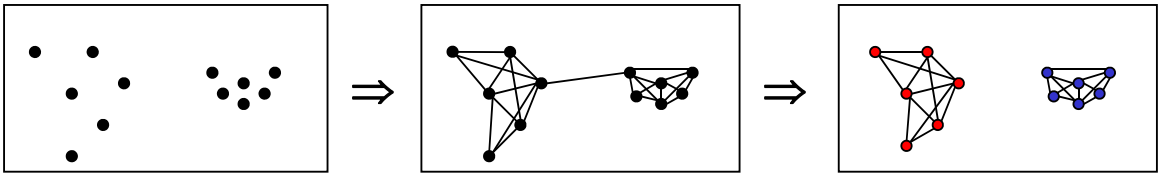


SNN_6-similarity(●,●) = 4

SNN_6-similarity(●,●) = 0

Einfaches SNN-Clustering [Jarvis, Patrick 73]:

1. Berechnung der Ähnlichkeitsmatrix und des Ähnlichkeitsgraphen
   - für alle Objekt-Paare $p,q \in DB$: berechne SNN_k-similarity(p,q)
   - SNN_k-Ähnlichkeitsgraph:
     - Knoten = Objekte
     - Kante zwischen jedem Objektpaar p,q mit Gewicht SNN_k-similarity(p,q)
   - Keine Kanten mit Gewicht 0
2. Generiere Cluster
   - Lösche alle Kanten, deren Gewicht unterhalb eines Grenzwerts τ liegen
   - Cluster = verbundene Komponenten im resultierenden Graphen

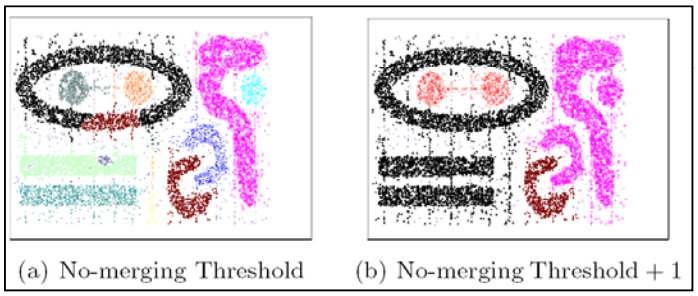


Datensatz | SNN_6 Ähnlichkeitsgraph | Generiere Cluster

5.3 Dichtebasiertes Clustering

Problem:

- Threshold τ schwer zu bestimmen
- kleine Variationen führen zu stark unterschiedlichen Ergebnissen



(a) No-merging Threshold (b) No-merging Threshold + 1

Bild aus:

[Ertöz, Steinbach, Kumar 03]

Lösung [Ertöz, Steinbach, Kumar 03]

- Kombiniere SNN-Ähnlichkeit mit dichtebasierten Konzepten
- SNN-Dichte:

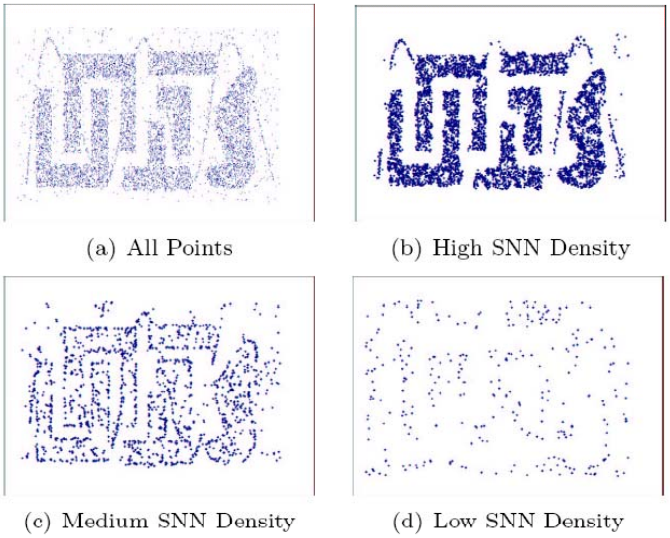
  Anzahl der Punkte innerhalb eines spezifizierten Radius ε bzgl. SNN-Ähnlichkeit

  $SNN_k\text{-density}(p,\varepsilon) = |\{q \mid SNN_k\text{-similarity}(p,q) \geq \varepsilon\}|$

5.3 Dichtebasiertes Clustering

SNN-Dichte

- Beispiel:
  - 10 000 Daten (Bild (a))
    $k = 50$, $\varepsilon = 20$
  - Bild (b): "Kernpunkte"
    alle Punkte mit SNN-Dichte ≥ 34
  - Bild (c): "Randpunkte"
    alle Punkte mit SNN-Dichte ≥ 17
  - Bild (d): "Rauschen"
    alle Punkte mit SNN-Dichte < 17



(a) All Points (b) High SNN Density

(c) Medium SNN Density (d) Low SNN Density

Bild aus: [Ertöz, Steinbach, Kumar 03]

- Analogie zu DBSCAN: $\varepsilon = 20$, *minPts* = 34
- Kernpunkt *p*: mehr als *minPts* Punkte haben 20 oder mehr der 50 nächsten Nachbarn mit *p* gemeinsam

SNN-Clustering Algorithmus [Ertöz, Steinbach, Kumar 03]

Eingabe: k, ε, *minPts*

1. Berechne Ähnlichkeitsmatrix und –graph
   (siehe einfaches SNN-Clustering)
2. Berechne die SNN_k-Dichte für jeden Punkt bzgl. ε
3. Bestimme Kernpunkte bzgl. *minPts*
   (alle Punkte mit einer SNN-Dichte $\geq$ *minPts*)
4. Vereinige Kernpunkte p,q, wenn SNN_k-similarity(p,q) $\geq \varepsilon$
5. Ordne Nicht-Kernpunkt p einem Cluster zu, wenn es einen Kernpunkt q gibt, mit SNN_k-similarity(q,p) $\geq \varepsilon$
6. Alle anderen Nicht-Kernpunkte sind Rauschen

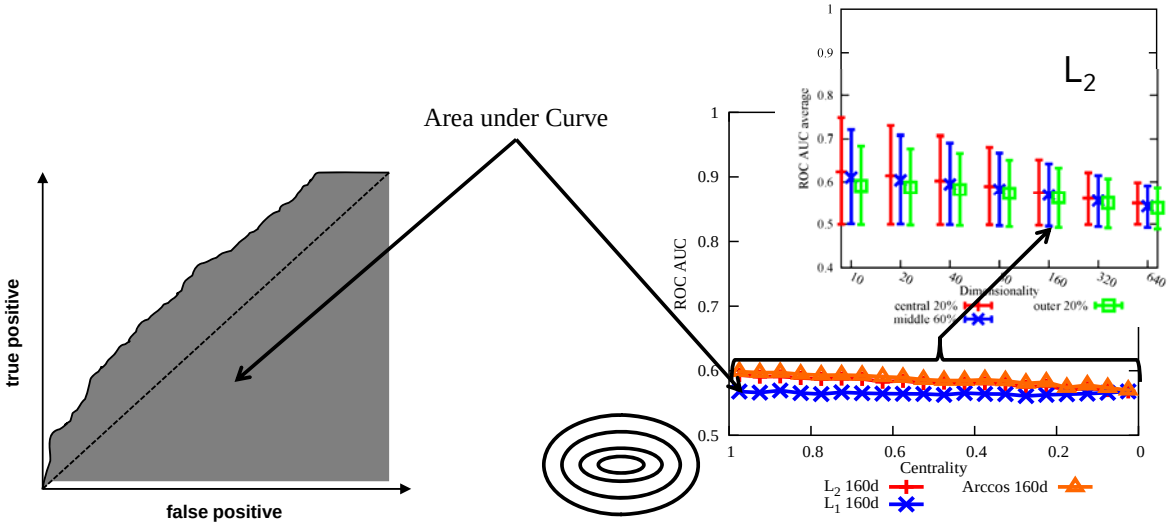
→ DBSCAN mit SNN-Ähnlichkeit

Diskussion

- Unterschied zu DBSCAN
  - DBSCAN mit Euklidischer Distanz: nur Cluster, die dichter sind als der Grenzwert (spezifiziert durch *minPts* und ε)
  - SNN-Dichte eines Punktes p: Anzahl der Punkte, die mind. ε nächste Nachbarn mit p gemeinsam haben
  - $\Rightarrow$ unabhängig von der eigentlichen Dichte
- Parametrisierung
  - Wahl von k ist kritisch:
    - Zu klein: auch relativ gleichverteilte Cluster werden wegen lokalen Variationen gesplittet $\Rightarrow$ viele kleine Cluster
    - Zu groß: wenige große, gut-separierte Cluster
  - *minPts*, $\varepsilon < k$

Studie: Stabilität von SNN [Houle, Kriegel, Kröger, Schubert, Zimek 2010]

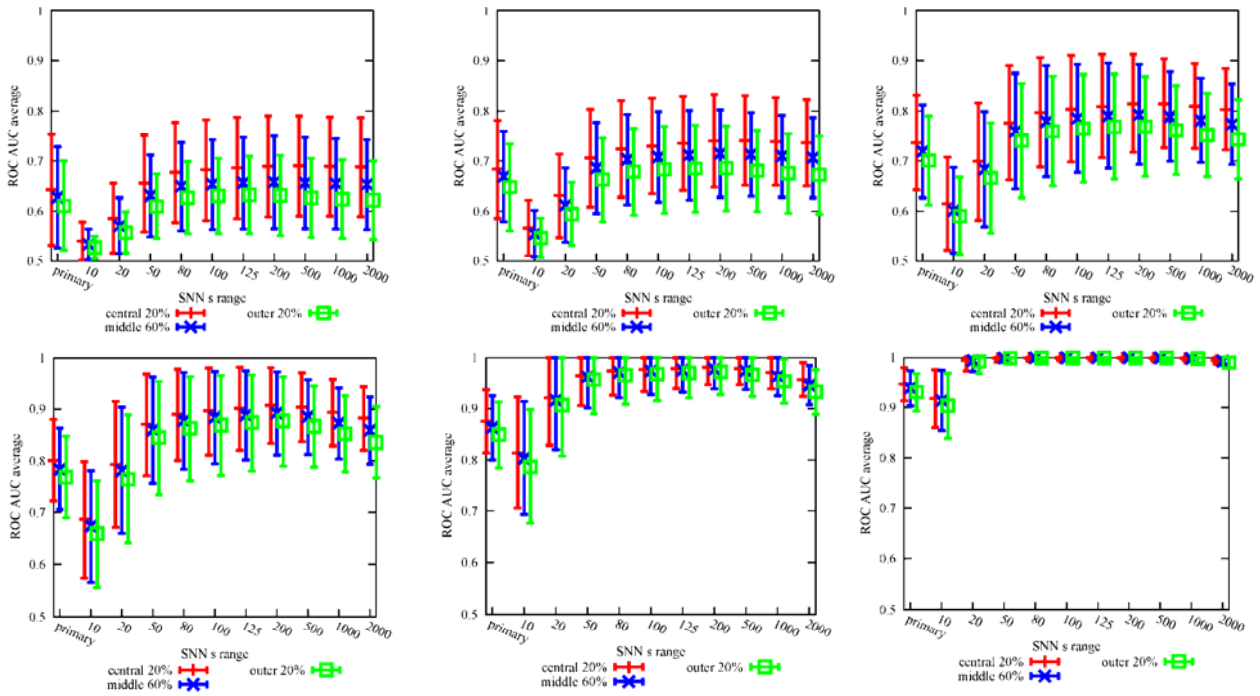
Experiment: Jedes Datenbankobjekt wird einmal als Anfrageobjekt verwendet, Nachbarschaftsranking: Area under curve (AUC) of the Receiver Operating Characteristic (ROC)



eine Anfrage (Nachbarn gleicher vs. anderer Cluster)

Durchschnitt aller Anfragen

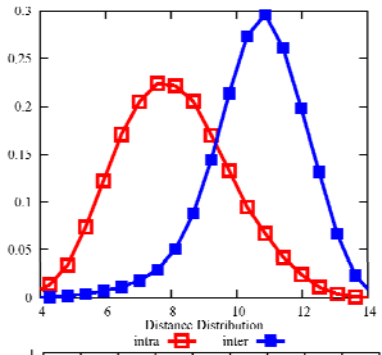
SNN basierend auf Euklidischer Distanz, künstliche Datensätze, alle Attribute relevant, 20/40/80/160/320/640 Dimensionen



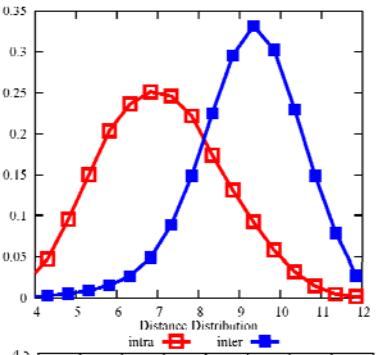
5.3 Dichtebasiertes Clustering

Verteilungen Euklidischer Distanz

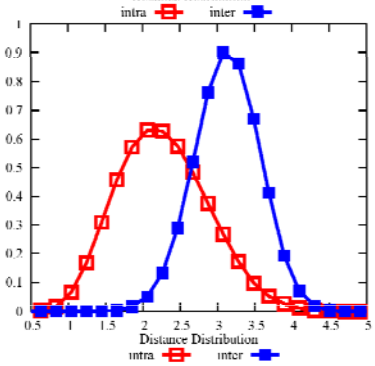
Multiple Features



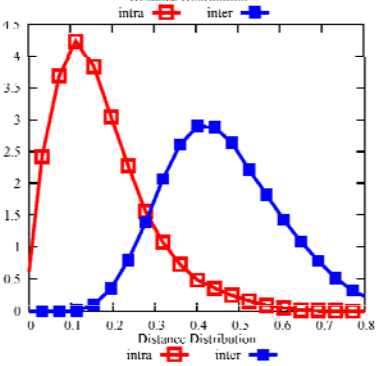
Multiple Features (pixel only)



Optical Digits



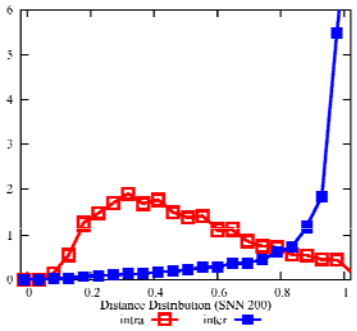
ALOI



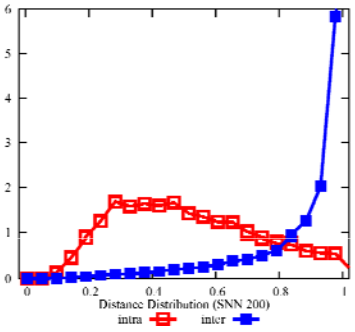
5.3 Dichtebasiertes Clustering

Verteilungen SNN Distanz

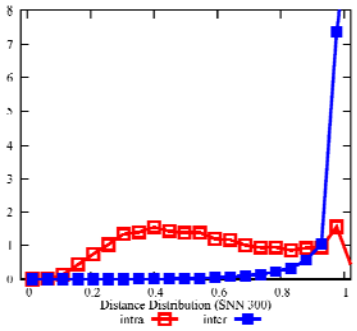
Multiple Features



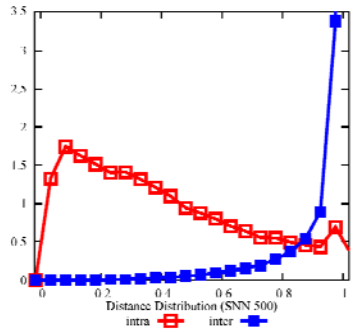
Multiple Features (pixel only)



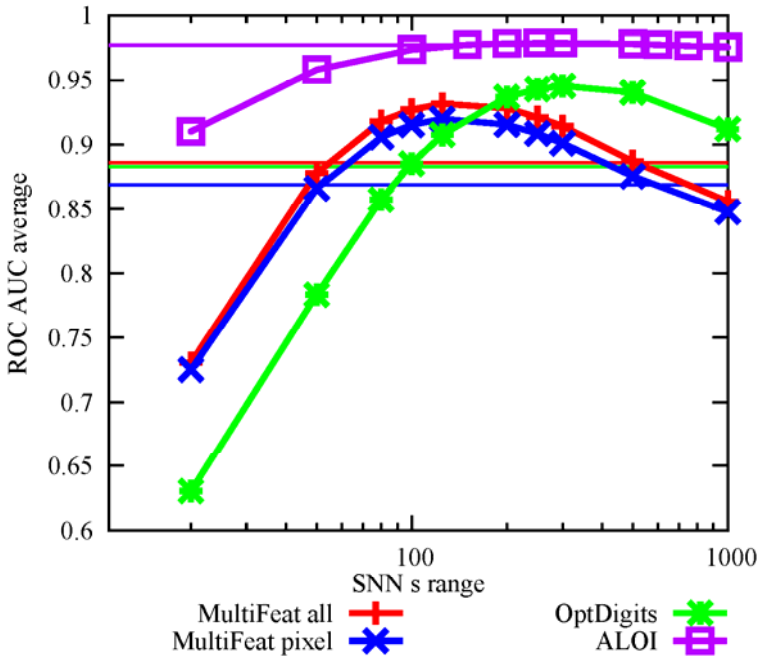
Optical Digits



ALOI



Ranking-Qualität



mehr Ergebnisse auf:

http://www.dbs.ifi.lmu.de/research/SNN/

Generalisierung von DBSCAN [Sander, Ester, Kriegel & Xu 1998]

Grundidee für dichte-basierte Cluster :

"*ε-Nachbarschaft* enthält mindestens *MinPts* Punkte"

"Distanz ≤ ε"

$NPred(o,p)$
reflexiv, symmetrisch
für Paare von Objekten

Verallgemeinerte Nachbarschaft
$N_{NPred}(o) = \{p \mid NPred(o, p)\}$

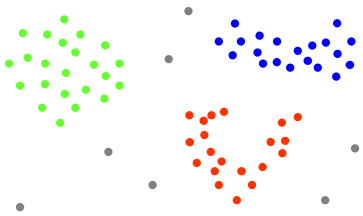
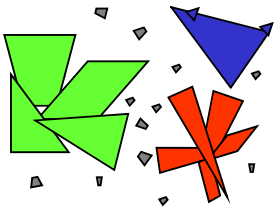
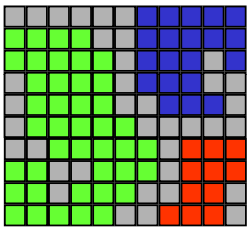
"$|N_\varepsilon| \geq MinPts$"

$MinWeight(N)$
beliebiges Prädikat für
Mengen von Objekten

Verallgemeinerte minimale Kardinalität
$MinWeight(N_{NPred}(o))$

"*NPred-Nachbarschaft* hat mindestens das "Gewicht" *MinWeight*"

Beispiele

			
NPred	dist(p,q) ≤ ε	intersect(p,q)	Nachbarzelle und ähnliche Farbe
MinWeight	cardinality(. . .) ≥ MinPoints	Summe der Flächen ≥ 5 % der Gesamtfläche	true

Algorithmus GDBSCAN

- dasselbe algorithmische Schema wie DBSCAN
- anstelle einer RQ(o,ε)-Anfrage eine N_{NPred}-Anfrage
- anstelle der Bedingung |RQ(o,ε)| ≥ *MinPts* das *MinWeight*-Prädikat auswerten
- Laufzeitkomplexität O(n log n) bei geeigneter Unterstützung der N_{NPred}-Anfrage
- Beliebige Nachbarschaftsprädikate denkbar