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

Skript zur Vorlesung

Knowledge Discovery in Databases

im Wintersemester 2010/2011

Kapitel 5: Clustering

Vorlesung+Übungen:
PD Dr. Peer Kröger, Dr. Arthur Zimek

Skript © 2003 Johannes Aßfalg, Christian Böhm, Karsten Borgwardt, Martin Ester, Eshref Januzaj, Karin Kailing, Peer Kröger, Jörg Sander, Matthias Schubert, Arthur Zimek

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

Ziel des Clustering

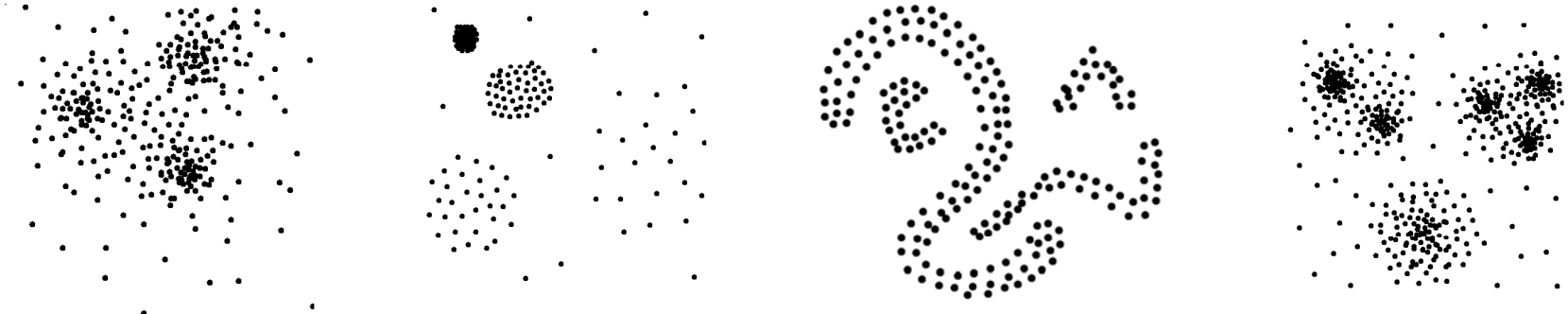
Identifikation einer endlichen Menge von Kategorien, Klassen oder Gruppen (*Cluster*) in den Daten.

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

5.1 Einleitung

- 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
- Dichtebasierte 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
  - …

5.2 Partitionierende Verfahren

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
- Diese führt meistens zu sphärisch geformten Clustern

5.2 Partitionierende Verfahren

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*)

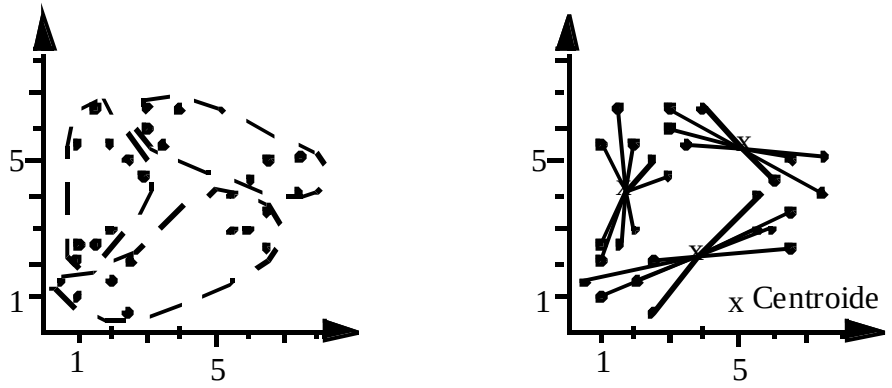
5.2 Partitionierende Verfahren

Konstruktion zentraler Punkte (Beispiel)

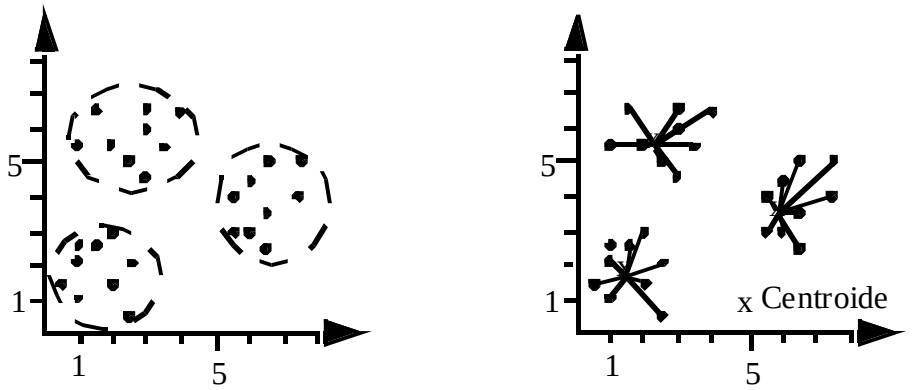
Cluster

Cluster-Repräsentanten

schlechtes Clustering



optimales Clustering



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*

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

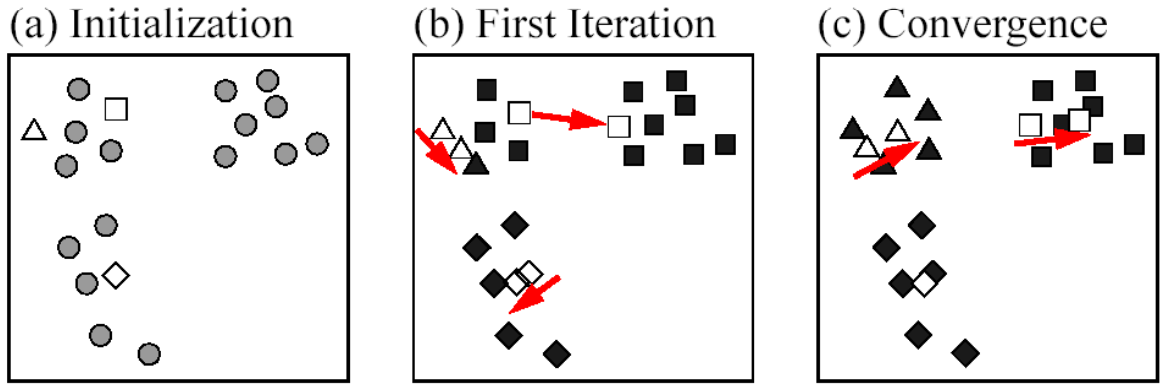
- *Maß für die Kosten* (Kompaktheit) *eines Clustering*

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

5.2 Partitionierende Verfahren

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



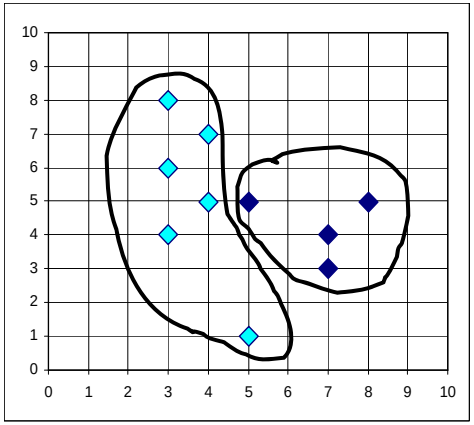
11

Algorithmus

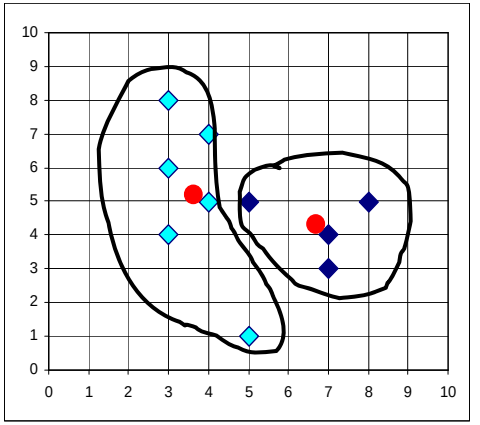
```
ClusteringDurchVarianzMinimierung(Punktmenge D, Integer k)
    Erzeuge eine „initiale“ Zerlegung der Punktmenge D in k
      Klassen;
    Berechne die Menge C'={C_1, ..., C_k} der Centroide für
      die k Klassen;
    C = {};
    repeat
        C = C';
        Bilde k Klassen durch Zuordnung jedes 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;
```

5.2 Partitionierende Verfahren

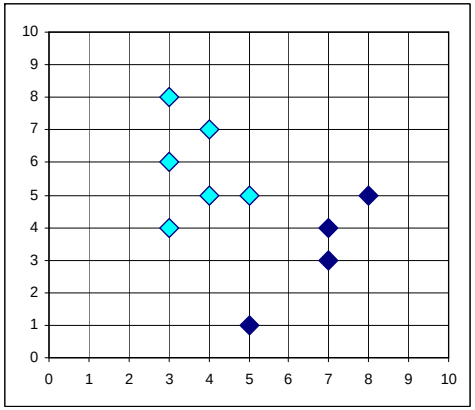
Beispiel



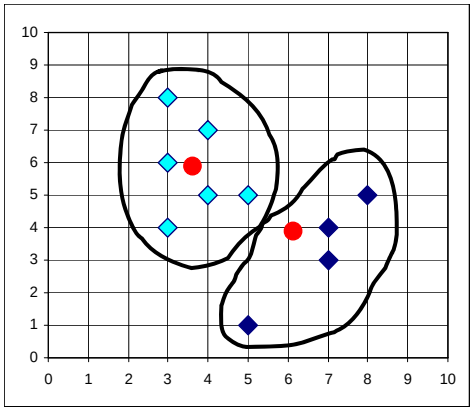
Berechnung der neuen Centroide



Zuordnung zum nächsten Centroid



Berechnung der neuen Centroide



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

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

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

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

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

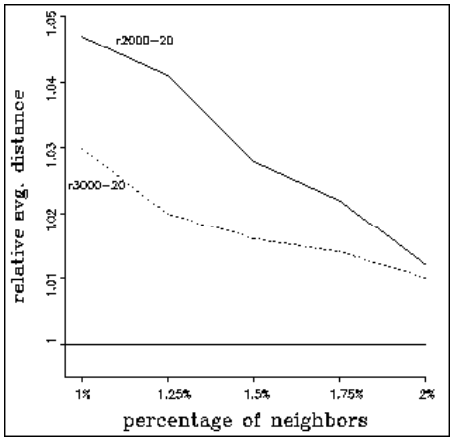
Laufzeitkomplexitäten

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

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

Experimentelle Untersuchung

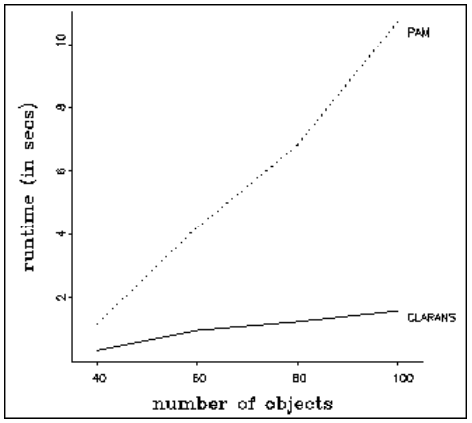
Qualität



TD(CLARANS)

TD(PAM)

Laufzeit



Erwartungsmaximierung (EM)

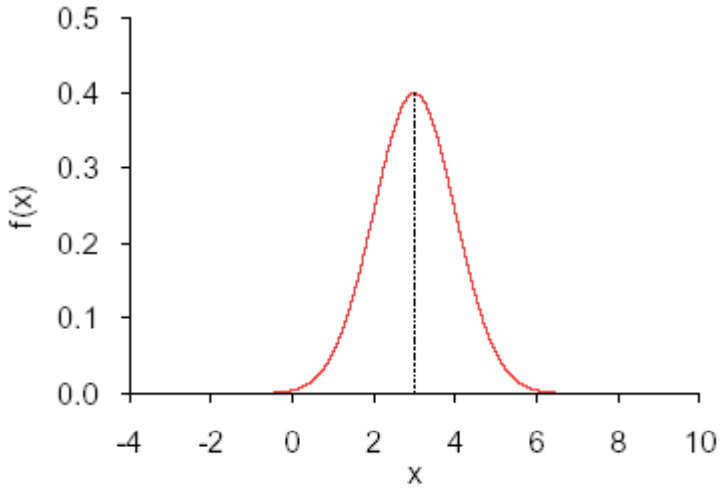
[Dempster, Laird & Rubin 1977]

- 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*

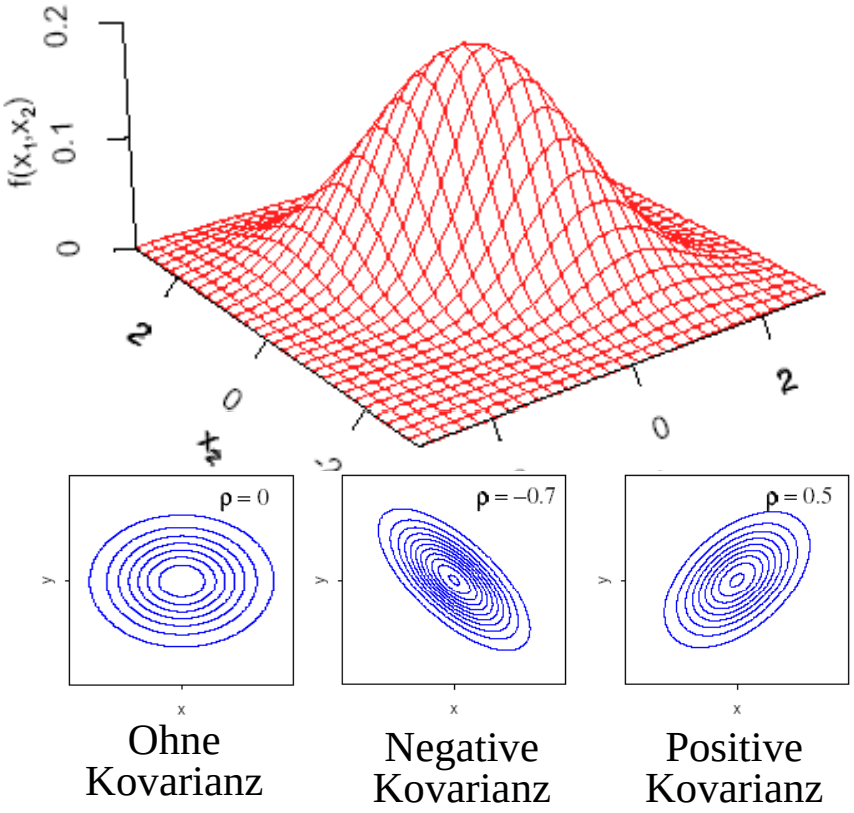
$$P(x \mid C) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_C \mid}} e^{-\frac{1}{2}\cdot(x-\mu_C)^T \cdot (\Sigma_C)^{-1} \cdot (x-\mu_C)}$$

Multivariate Normalverteilung

Univariate Normalverteilung



Bivariate Normalverteilung



Ohne Kovarianz

Negative Kovarianz

Positive Kovarianz

5.2 Partitionierende Verfahren

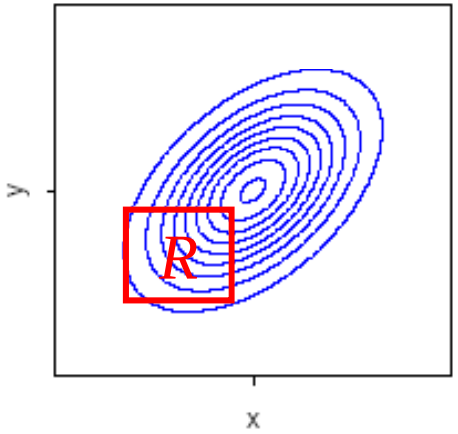
Idee des EM-Algorithmus:

- 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 Wahrscheinlichkeitsdichte 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

5.2 Partitionierende Verfahren

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

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

5.2 Partitionierende Verfahren

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

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

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 $P(C_i|x)$

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

5.2 Partitionierende Verfahren

- Maß für die Güte eines Clustering *M*

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

$\Longrightarrow$ *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:

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

$$\Sigma_i = \left(\sum_{x \in D} (x - \mu_i)(x - \mu_i)^{\mathrm{T}} \cdot P(C_i \mid x) \right) / \left(\sum_{x \in D} P(C_i \mid x) \right)$$

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.