

9 Automatische Klassifikation

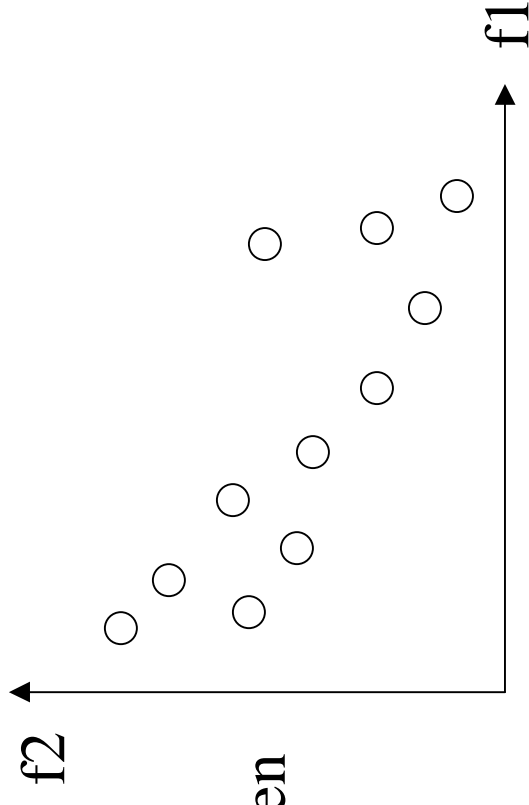
Ziel:

Organisation von Dokumenten in (hierarchischen) Ontologien mit möglichst geringem intellektuellem Aufwand

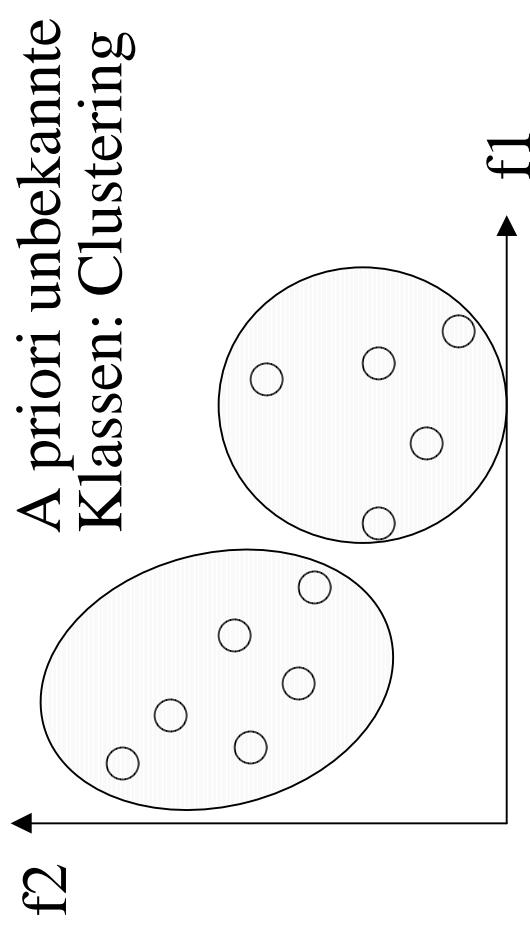
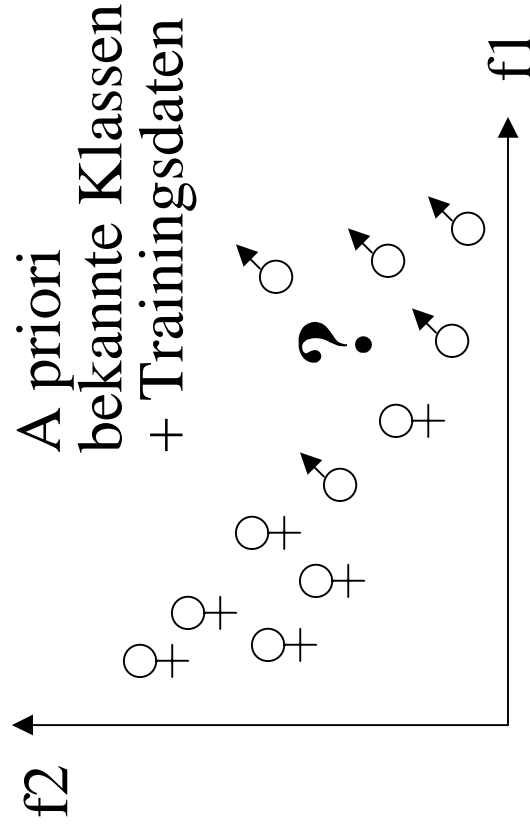
- 9.1 Klassifikationsverfahren mit Training
- 9.2 Clustering: Klassifikation ohne Training
- 9.3 Hypertext-Klassifikation
- 9.4 Hierarchische Klassifikation

Klassifikationsproblem (Kategorisierung)

gegeben:
Featurevektoren



bestimme
Klassenzugehörigkeit
von Feature-Vektoren



Anwendung von Klassifikationsverfahren im IR

- *Filtern*: teste eintreffende Dokumente (z.B. Mail, News), ob sie in eine interessante Klasse fallen
- *Übersicht*: organisiere Query-/Crawler-Resultate, Verzeichnisse, Feeds, etc.
- *Query-Expansion*: ordne Query einer Klasse zu und ergänze dementsprechende Suchterme
- *Relevanz-Feedback*: klassifiziere Treffer und lasse Benutzer relevante Klassen identifizieren, um bessere Query zu generieren
- *Query-Effizienz*: beschränke (Index-)Suche auf relevante Klasse(n)

Klassifikationsvarianten:

- mit Termen, Termhäufigkeiten, Linkstruktur als Features
- binär: Gehört ein Dokument zu einer Klasse c oder nicht?
- mehrstellig: In welche von k Klassen passt ein Dokument am besten?
- hierarchisch: Iteration der Klassifikation über ontologischen Baum

Bewertung der Klassifikationsgüte

empirisch durch automatische Klassifikation von Dokumenten, die nicht zu den Trainingsdaten gehören

Für binäre Klassifikation bzgl. Klasse C:

a = #Dok., die zu C klassifiziert wurden und zu C gehören

b = #Dok., die zu C klassifiziert wurden, aber nicht zu C gehören

c = #Dok., die nicht zu C klassifiziert wurden, aber zu C gehören

d = #Dok., die nicht zu C klassifiziert wurden und nicht zu C gehören

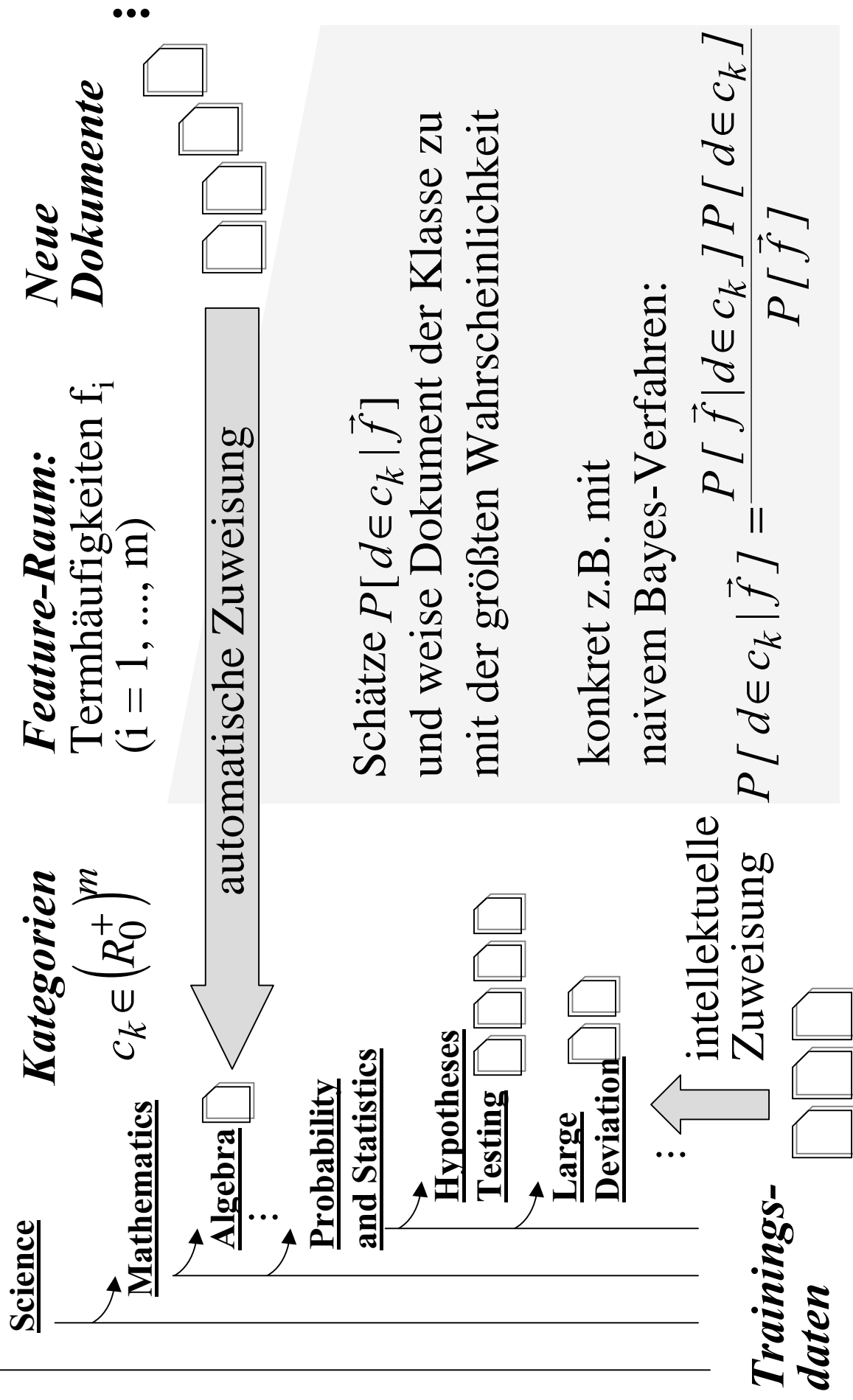
$$\text{Genauigkeit (accuracy)} = \frac{a + d}{a + b + c + d}$$

$$\text{Präzision (precision)} = \frac{a}{a + b} \quad \text{Ausbeute (recall)} = \frac{a}{a + c}$$

Für mehrstellige Klassifikation bzgl. Klassen C1, ..., Ck:

- Makrodurchschnitt über k Klassen oder
- Mikrodurchschnitt über k Klassen

9.1 Klassifikationsverfahren mit Training (supervised learning)



k-Nearest-Neighbor-Verfahren (kNN)

Schritt 1:

Finde unter den Trainingsdokumenten aller Klassen die k (z.B. 10-100) bzgl. der (Cosinus-) Ähnlichkeit nächsten Nachbarn eines neuen Dokuments \vec{d}

Schritt 2:

Ordne \vec{d} derjenigen Klasse C_j zu, für die die Funktion

$$f(\vec{d}, C_j) = \sum_{\vec{v} \in kNN(\vec{d})} \text{sim}(\vec{d}, \vec{v}) * \begin{cases} 1 & \text{falls } \vec{v} \in C_j \\ 0 & \text{sonst} \end{cases}$$

maximal wird

Bei binärer Klassifikation ordne \vec{d} der Klasse C zu, falls $f(\vec{d}, C)$ über einem Schwellwert δ ($\delta > 0.5$) liegt.

Klassifikationsverfahren von Rocchio

Schritt 1:

Repräsentiere die Trainingsdokumente einer Klasse C_j

- mit tf*idf-Vektorkomponenten – durch den **Prototypvektor**:

$$\vec{c}_j := \alpha \frac{1}{|C_j|} \sum_{\vec{d} \in C_j} \frac{\vec{d}}{\|\vec{d}\|} - \beta \frac{1}{|D - C_j|} \sum_{\vec{d} \in D - C_j} \frac{\vec{d}}{\|\vec{d}\|}$$

mit geeigneten Koeffizienten α und β (z.B. $\alpha=16$, $\beta=4$)

Schritt 2:

Ordne ein neues Dokument \vec{d} derjenigen Klasse C_j zu, für die Cosinus-Ähnlichkeit $\cos(\vec{c}_j, \vec{d})$ maximal ist.

Satz:

Für $\alpha=\beta=1$ maximiert \vec{c}_j die Funktion:

$$f(\vec{c}_j) = \alpha \frac{1}{|C_j|} \sum_{\vec{d} \in C_j} \cos(\vec{c}_j, \vec{d}) - \frac{1}{|D - C_j|} \sum_{\vec{d} \in D - C_j} \cos(\vec{c}_j, \vec{d})$$

Naives Bayes-Verfahren mit binären Features Xi

$$\text{Schätze: } P[d \in c_k | d \text{ hat } \vec{X}] = \frac{P[d \text{ hat } \vec{X} | d \in c_k] P[d \in c_k]}{P[d \text{ hat } \vec{X}]}$$

$$\sim P[X | d \in c_k] P[d \in c_k]$$

$$= \prod_{i=1}^m P[X_i | d \in c_k] P[d \in c_k]$$

bei Featureunabhängigkeit
bzw. Linked Dependence:

$$\frac{P[X | d \in c_k]}{P[X | d \notin c_k]} = \prod_i \frac{P[X_i | d \in c_k]}{P[X_i | d \notin c_k]}$$

$$= \prod_{i=1}^m p_{ik}^{X_i} (1 - p_{ik})^{1-X_i} p_k$$

mit empirisch zu schätzenden
 $p_{ik} = P[X_i = 1 | c_k], p_k = P[c_k]$

$$\Rightarrow \log P[c_k | d] \sim \sum_{i=1}^m X_i \log \frac{p_{ik}}{(1 - p_{ik})} + \sum_{i=1}^m \log(1 - p_{ik}) + \log p_k$$

für binäre Klassifikation mit Quote statt P[...] weitere Vereinfachung

Naives Bayes-Verfahren mit Bag-of-Words-Modell

Schätze: $P[d \in c_k | d \text{ hat } \vec{f}] \sim P[\vec{f} | d \in c_k] P[d \in c_k]$

mit Termhäufigkeitsvektor \vec{f}

$= \prod_{i=1}^m P[f_i | d \in c_k] P[d \in c_k]$ bei Featureunabhängigkeit

$$= \prod_{i=1}^m \binom{\text{length}(d)}{f_i} p_{ik}^{f_i} (1 - p_{ik})^{\text{length}(d) - f_i} p_k$$

mit Binomialverteilung
für jedes Feature

bzw.
besser:

$$= \binom{\text{length}(d)}{f_1 f_2 \dots f_m} p_{1k}^{f_1} p_{2k}^{f_2} \dots p_{mk}^{f_m} p_k$$

mit Multinomialverteilung
der Featurevektoren und

mit $\binom{n}{k_1 k_2 \dots k_m} := \frac{n!}{k_1! k_2! \dots k_m!}$

$$\sum_{i=1}^m f_i = \text{length}(d)$$

Beispiel für das naive Bayes-Verfahren (1)

3 Klassen: c1 – Algebra, c2 – Analysis, c3 – Stochastik

8 Terme, 6 Trainingsdokumente d1, ..., d6: je 2 in jeder Klasse

$\Rightarrow p1=2/6, p2=2/6, p3=2/6$

		Gruppe								Homomorphismus				Integral				Limes				Varianz				Wahrscheinlichkeit				Würfel				Algebra				Analysis				Stochastik																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
		f1	f2	f3	f4	f5	f6	f7	f8																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								

Beispiel für das naive Bayes-Verfahren (2)

Klassifikation von d7: (0 0 1 2 0 0 3 0)

$$P[\vec{f} | d \in c_k] P[d \in c_k] = \binom{\text{length}(d)}{f_1 f_2 \dots f_m} p_{1k}^{f_1} p_{2k}^{f_2} \dots p_{mk}^{f_m} p_k$$

$$\text{für } k=1 \text{ (Algebra):} \quad = \binom{6}{1 \ 2 \ 3} \left(\frac{3}{12}\right)^1 0^2 0^3 \frac{2}{6} = 0$$

$$\text{für } k=2 \text{ (Analysis):} \quad = \binom{6}{1 \ 2 \ 3} \left(\frac{1}{12}\right)^1 \left(\frac{5}{12}\right)^2 \left(\frac{1}{12}\right)^3 \frac{2}{6} = 20 * \frac{25}{12^6}$$

$$\text{für } k=3 \text{ (Stochastik):} = \binom{6}{1 \ 2 \ 3} \left(\frac{1}{12}\right)^1 \left(\frac{1}{12}\right)^2 \left(\frac{4}{12}\right)^3 \frac{2}{6} = 20 * \frac{64}{12^6}$$

Resultat: Ordne d7 der Klasse C3 (Stochastik) zu

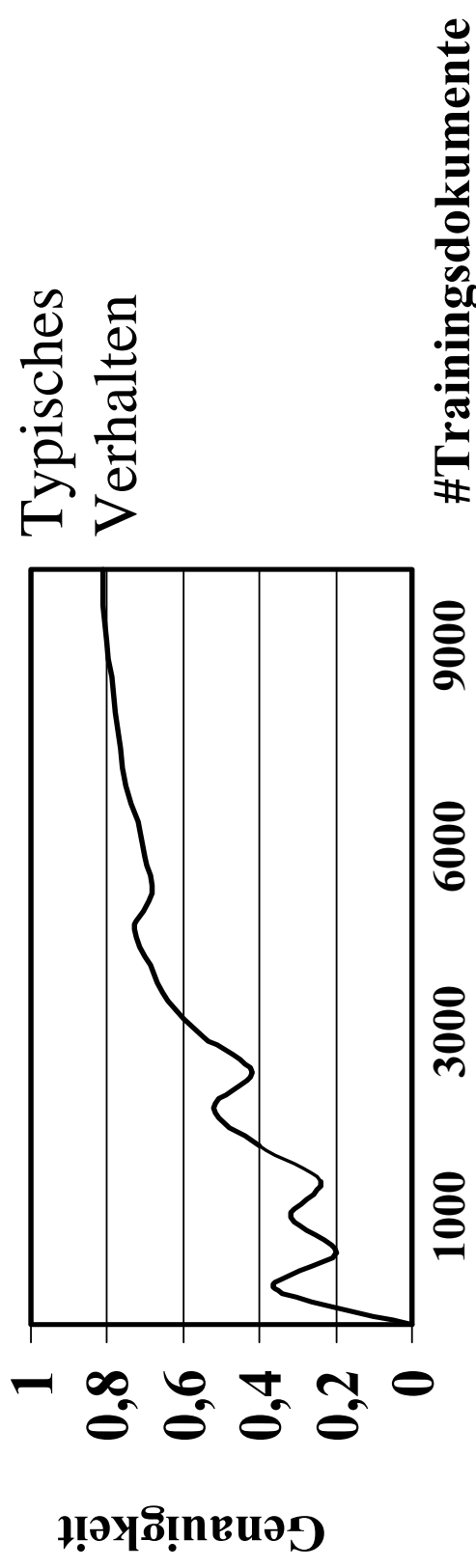
Typisches Verhalten des naiven Bayes-Verfahrens

Reuters Benchmark (siehe trec.nist.gov):

12902 kurze Artikel (Wirtschaftsnachrichten)

aus 90 Kategorien (acq, corn, earn, grain, interest, money-fx, ship, ...)

- Verwende die (bzw. einen Teil der) ältesten 9603 Artikel zum Trainieren des Klassifikators
- Verwende die neuesten 3299 Artikel zur Evaluation der Klassifikationsgenauigkeit



max. Genauigkeit liegt je nach Kategorie zwischen 50 und 90 Prozent

Verbesserung des naiven Bayes-Verfahrens

1) geglättete Schätzung der pik (Laplace smoothing):

$$\frac{1}{\sum_{d \in C_k} \text{length}(d)} \quad \text{statt } 0 \quad \text{für in den Trainingsdokumenten einer Klasse überhaupt nicht auftretende Features}$$

2) Anreicherung des Trainingsmaterials durch unbenannte, automatisch klassifizierte Dokumente zur besseren Schätzung der pik

mit unterschiedlicher Gewichtung der intellektuell und der automatisch klassifizierten „Trainingsdokumente“

3) Berücksichtigung von Abhängigkeiten zwischen Features durch Verallgemeinerung auf Bayessche Netze

Entscheidungsbäume

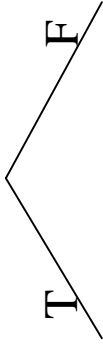
gegeben: eine Anzahl m -dimensionaler Trainingsdatensätze
 $\subseteq \text{dom}(A_1) \times \dots \times \text{dom}(A_m)$ mit
numerischen, ordinalen oder kategorialen Attributen A_i
(z.B. Termhäufigkeitsvektoren $\subseteq \mathbb{N}_0 \times \dots \times \mathbb{N}_0$)

gesucht: ein Baum mit

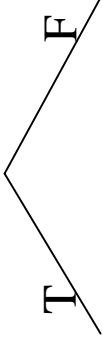
- Attributwertbedingungen der Form
 - $A_i \leq \text{Wert}$ für numerische und ordinale Attribute
 - oder
 - $A_i \in \text{Wertemenge}$ bzw. $A_i \cap \text{Wertemenge} = \emptyset$
für kategoriale Attribute
 - oder
 - Linearkombinationen der Form $\sum k_i A_i \leq \text{Wert}$
über mehrere numerische Attribute
- als inneren Knoten und
- benannten Klassen als Blättern

Beispiele für Entscheidungsbäume (1)

$\text{tf}(\text{Homomorphismus}) \geq 2$



hat Tolkien gelesen



$\text{tf}(\text{Vektor}) \geq 3$ $\text{tf}(\text{Limes}) \geq 2$



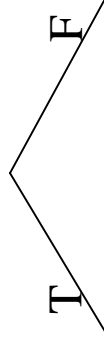
hat Eco gelesen *Fantasielose*



*Lineare Algebra Analysis Sonst.
Algebra*

Intellektuelle Bananen

Gehalt ≥ 100000



kreditwürdig

Hochschulabschluß &

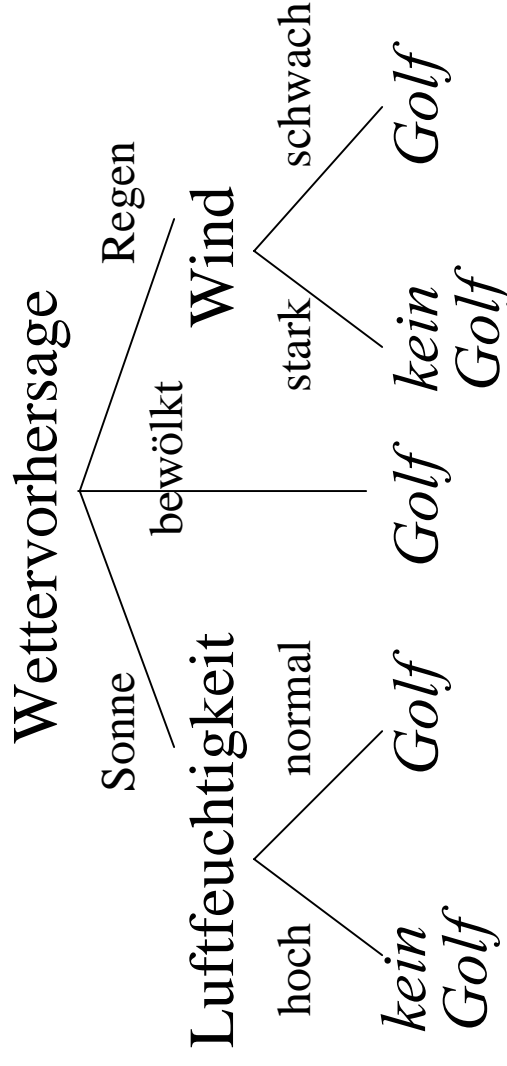
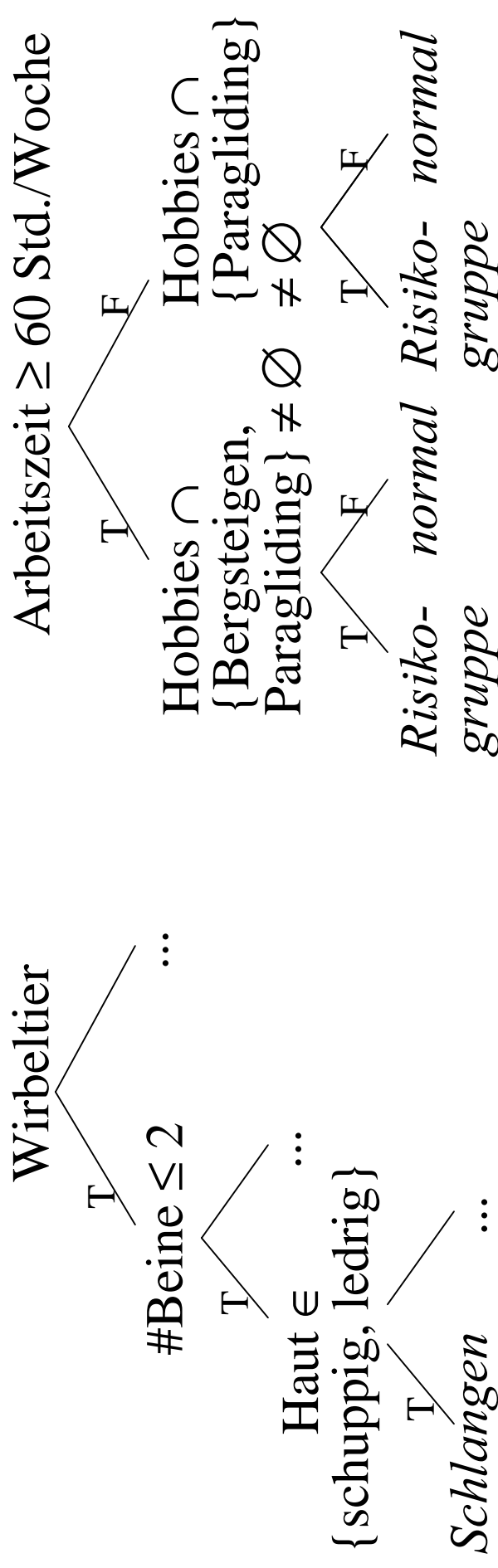
Gehalt ≥ 50000



kreditwürdig

nicht kreditwürdig

Beispiele für Entscheidungsbäume (2)



Top-Down-Aufbau eines Entscheidungsbaums

Input: Entscheidungbaumknoten k , der eine

Partition D von $\text{dom}(A_1) \times \dots \times \text{dom}(A_m)$ beschreibt

Output: Entscheidungsbaum mit k als Wurzel

- 1) BuildTree (root, $\text{dom}(A_1) \times \dots \times \text{dom}(A_m)$)
- 2) PruneTree: Stutze Baum auf günstigste Größe

mit:

procedure BuildTree (k, D):

Falls k nur Trainingsdaten derselben Klasse enthält, terminiere

Bestimme Splitdimension A_i

Bestimme Splitwert x , um D möglichst günstig in

$D_1 = D \cap \{d \mid d.A_i \leq x\}$ und $D_2 = D \cap \{d \mid d.A_i > x\}$ aufzuteilen

Erzeuge Kinder k_1 und k_2 von k

BuildTree (k_1, D_1); BuildTree (k_2, D_2)

Splitkriterium Informationsgewinn

Ziel ist es, so zu splitten, daß bzgl. der Klassen der Trainingsdaten möglichst reine Partitionen entstehen, daß also die Unreinheit (impurity) der Partitionierung minimiert wird.

Eine Art, Reinheit zu definieren, ist durch den mittels **Entropie** definierten (statistischen) **Informationsgewinn** (information gain):

$$G(k, k1, k2) = H(k) - (p1 * H(k1) + p2 * H(k2))$$

Dabei sind:

n_k : Anzahl der Trainingsdatensätze in k

$n_{k,j}$: Anzahl der Trainingsdatensätze in k , die zur Klasse j gehören

$$p1 = n_{k1} / n_k \text{ und } p2 = n_{k2} / n_k$$

$$H(k) = - \sum_j \frac{n_{k,j}}{n_k} \log_2 \frac{n_{k,j}}{n_k}$$

Alternative Splitkriterien

1) Split, so daß Entropie von k_1 und k_2 minimiert wird:

$$p_1 * H(k_1) + p_2 * H(k_2)$$

2) Split, so daß $GI(k_1) + GI(k_2)$ minimiert wird
mit dem „Gini-Index“:

$$GI(k) = 1 - \sum_j \left(\frac{n_{k,j}}{n_k} \right)^2$$

3) Verzweigungen nach Attributen mit vielen Werten
werden vom Kriterium des Informationsgewinns bevorzugt
Abhilfe:

Splitkriterium Informationsgewinnverhältnis (gain ratio)

$$G(k, k_1, k_2) / H(k)$$

Kriterien für das Stutzen des Baums

Problem: Zu vollständige Entscheidungsbäume tendieren zum „**Overfitting**“ – Verzweigungen trotz wenig aussagekräftiger („verrauschter“) Daten:

Minimierung des Klassifikationsfehlers auf Trainingsdaten, aber Verschlechterung gegenüber kleinerem Baum auf Testdaten

Lösung: Entferne Blätter, bis nur noch aussagekräftige Knoten übrig sind, z.B. nach dem Prinzip der

Minimum Description Length (MDL):

beschreibe die Klassenzugehörigkeit aller Trainingsdaten

mit minimaler Länge (in Bits)

- K Bits pro Knoten des Baums (Attribut, Attributwert, Zeiger)
- $n_k * H(k)$ Bits für die explizite Klassenzuordnung aller n_k Datensätze eines Blattknotens k mit

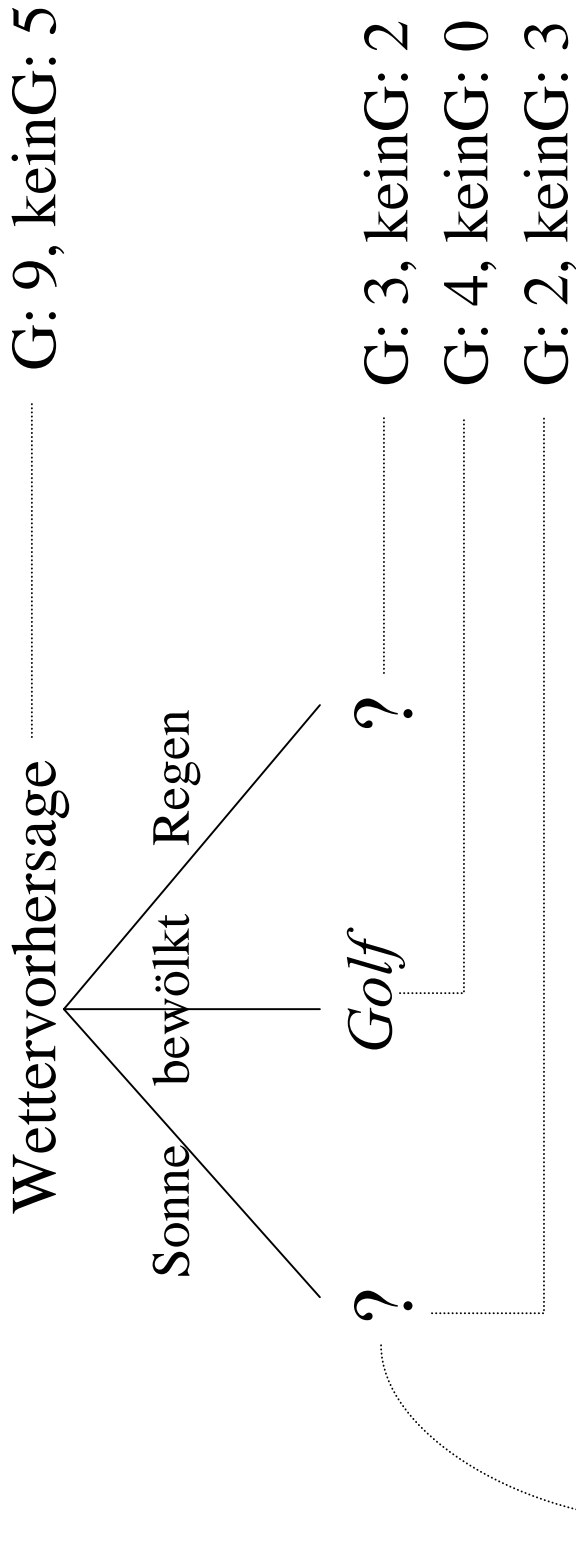
$$H(k) = - \sum_j \frac{n_{k,j}}{n_k} \log_2 \frac{n_{k,j}}{n_k}$$

Beispiel für Entscheidungsbaumverfahren (1)

Trainingsdaten:

	Wetter- vorhersage	Temperatur	Luft- feuchtigkeit	Wind	Golf
1)	Sonne	heiß	hoch	schwach	nein
2)	Sonne	heiß	hoch	stark	nein
3)	bewölkt	heiß	hoch	schwach	ja
4)	Regen	mild	hoch	schwach	ja
5)	Regen	kalt	normal	schwach	ja
6)	Regen	kalt	normal	stark	nein
7)	bewölkt	kalt	normal	stark	ja
8)	Sonne	mild	hoch	schwach	nein
9)	Sonne	kalt	normal	schwach	ja
10)	Regen	mild	normal	schwach	ja
11)	Sonne	mild	normal	stark	ja
12)	bewölkt	mild	hoch	stark	ja
13)	bewölkt	heiß	normal	schwach	ja
14)	Regen	mild	hoch	stark	nein

Beispiel für Entscheidungsbaumverfahren (2)



Datensätze: 1, 2, 8, 9, 11

Entropie $H(k)$: $2/5 * \log_2 5/2 + 3/5 * \log_2 5/3 \approx 2/5 * 1.32 + 3/5 * 0.73 \approx 0.970$

Wahl des Splitattributs:

G(Luftfeuchtigkeit): $0.970 - 3/5 * 0 - 2/5 * 0 = 0.970$

G(Temperatur): $0.970 - 2/5 * 0 - 2/5 * 1 - 1/5 * 0 = 0.570$

G(Wind): $0.970 - 2/5 * 1 - 3/5 * 0.918 = 0.019$

Beispiel für Entscheidungsbaumverfahren zur Textklassifikation

	Gruppe	Homom.	Vektor	Integral	Limes	Varianz	Wahrsch.	Würfel	
C1: Algebra	d1:	f1	f2	f3	f4	f5	f6	f7	f8
	d2:	3	2	0	0	0	0	0	1
C2: Analysis	d3:	1	2	3	0	0	0	0	0
	d4:	0	0	0	3	3	0	0	0
C3: Stochastik	d5:	0	0	1	2	2	0	1	0
	d6:	0	0	0	1	1	2	2	0
		1	0	1	0	0	0	2	2

f2>0 -----

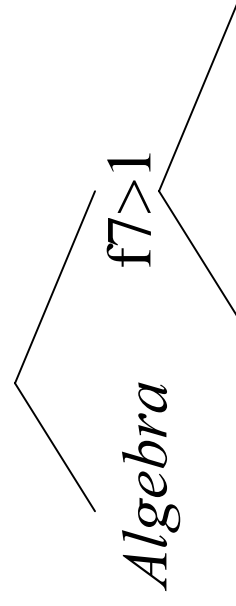
$$G = H(k) - (2/6 * H(k1) + 4/6 * H(k2))$$

$$H(k) = 1/3 \log 3 + 1/3 \log 3 + 1/3 \log 3$$

$$H(k1) = 1 \log 1 + 0 + 0$$

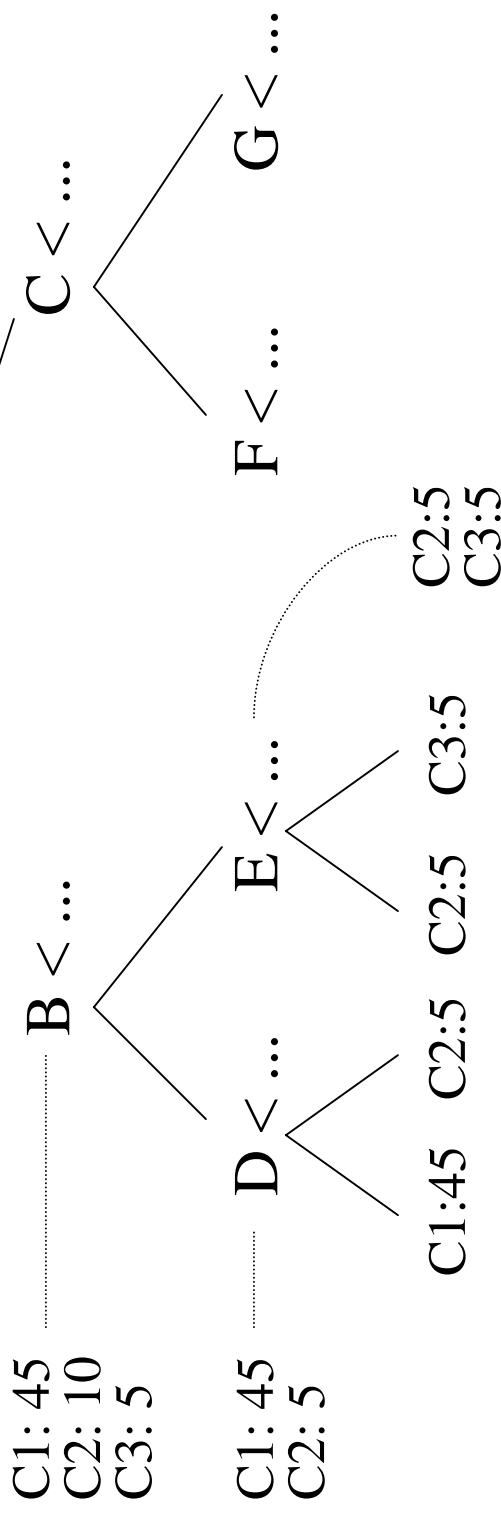
$$H(k2) = 0 + 1/2 \log 2 + 1/2 \log 2$$

$$G = \log 3 - 0 - 2/3 * 1 \approx 1,6 - 0,66 = 0,94$$



Beispiel für das Stutzen eines Entscheidungsbaums

3 Klassen: C1, C2, C3
 100 Trainingsdatensätze
 C1: 60, C2: 30, C3: 10



Annahme: Codierungskosten eines Baumknotens $K=30$ Bits

Codierung D-Teilbaum: $47 \cdot (0.9 \log_2 10/9 + 0.1 \log_2 10) \approx$

$$47 \cdot (0.9 \cdot 0.15 + 0.1 \cdot 3.3) \approx 47 \cdot 0.465 < 30$$

Codierung E-Teilbaum: $10 \cdot (0.5 \log_2 2 + 0.5 \log_2 2) = 10 < 30$

Codierung B-Teilbaum: $60 \cdot (9/12 \log_2 12/9 + 1/6 \log_2 6 + 1/12 \log_2 12) \approx$
 $60 \cdot (0.75 \cdot 0.4 + 0.166 \cdot 2.6 + 0.083 \cdot 3.6) > 30$

Probleme von Entscheidungsbaumverfahren für die Klassifikation von Textdokumenten

- Der Trainingsaufwand ist sehr hoch.
- Bei hochdimensionalen, dünnbesetzten Featurevektoren führen Trainingsdaten leicht zu „Overfitting“.

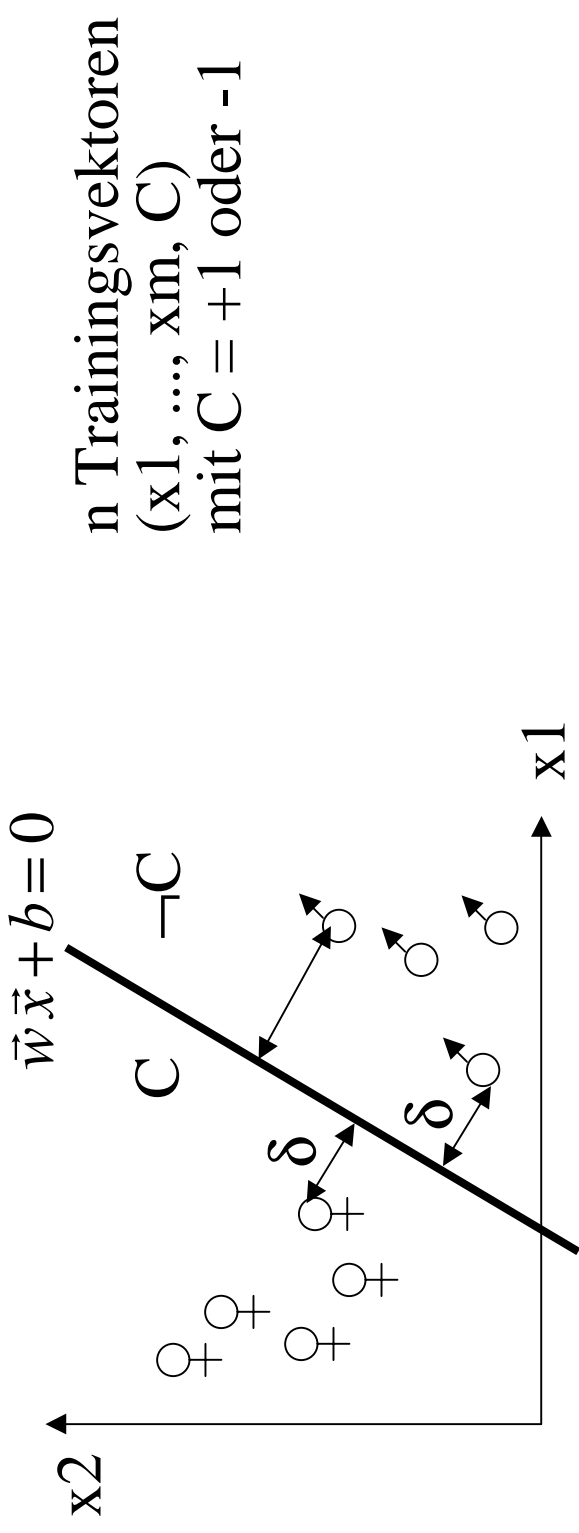
Entropiemaximierungsverfahren (Maximum Entropy Classifier)

Idee zur Schätzung von $P[d \in c_k \text{ und } d \text{ hat } \vec{f}]$:

- Bestimme die Parameter einer parametrisierten W. verteilung, die
- bzgl. der Erwartungswerte E_{ik} für alle Features f_i und Klassen C_k mit den empirischen Mittelwerten M_{ik} (aufgrund von n Trainingsvektoren) übereinstimmen und
 - maximale Entropie haben (d.h. Gleichverteilung postulieren, wenn die Trainingsdaten nichts Anderes aussagen)
- Verteilung hat die loglineare Form $P[C_k, \vec{f}] = \frac{1}{Z} \prod_i \alpha_i^{f_{ik}}$ mit einer Normalisierungskonstanten Z

Berechnung der Parameter α_i durch Iterationsverfahren (generalized iterative scaling), das unter bestimmten Voraussetzungen garantiert konvergiert

Stützvektorverfahren zur binären Klassifikation (Support Vector Machine = SVM)



Bestimme Hyperebene $\vec{w}\vec{x} + b = 0$, die Trainingsvektoren aus C von denen, die nicht aus C sind, optimal separiert, so daß der (Euklidische) Mindestabstand δ zu allen Trainingsvektoren maximiert wird. (Vektoren mit Abstand δ heißen Stützvektoren.)

Klassifiziere neuen Vektor \vec{y} in C durch Test: $(\vec{w}\vec{y} + b) = \sum_{i=1}^m w_i y_i + b > 0$

Berechnung der optimalen Hyperebene

Finde $\vec{w} \in R^m$ und $b \in R$, so daß

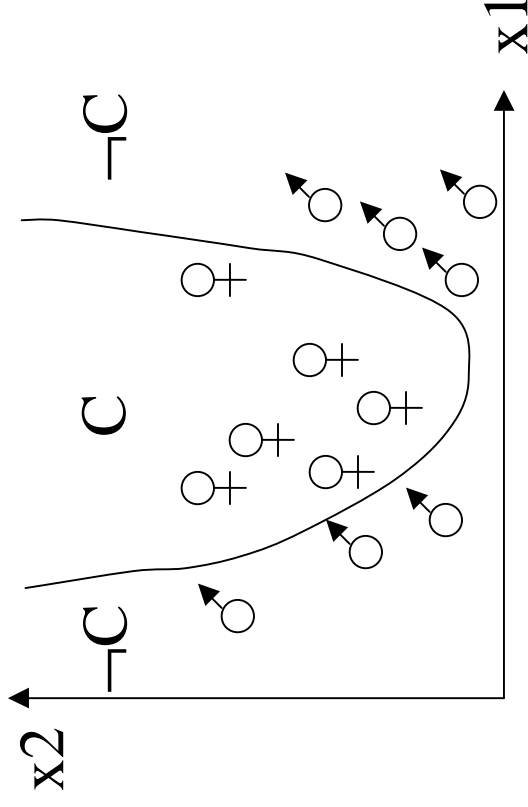
1. $\delta \in R$ maximal ist und
2. $C_i \frac{1}{\|\vec{w}\|} (\vec{w} \vec{x}_i + b) \geq \delta$ für alle $i=1, \dots, n$

Dies ist (o.B.d.A. mit der Wahl von $\|\vec{w}\|=1/\delta$) äquivalent zu
(V. Vapnik: Statistical Learning Theory, 1998):

- Finde $\alpha_1, \dots, \alpha_n \in R_0^+$, so daß
1. $\sum_{i=1}^n \alpha_i \frac{1}{2} \sum_{j=1}^n C_j C_i \alpha_j (\vec{x}_i \vec{x}_j)$ minimal ist (Problem der quadratischen Programmierung)
 2. und $\sum_{i=1}^n C_i \alpha_i = 0$

Optimaler Gewichtsvektor \vec{w} ist Linearkombination $\vec{w} = \sum_{i=1}^n \alpha_i C_i \vec{x}_i$
(wobei nur für Stützvektoren $\alpha_i > 0$)
b ergibt sich aus beliebigem Stützvektor \vec{x}_j durch: $b = C_j - \vec{w} \vec{x}_j$

Erweiterte SVMs zur nichtlinearen Klassifikation



Transformiere Vektoren $\vec{x} \in R^m$ auf $\Phi(\vec{x}) \in R^{m'}$ mit $m' > m$

z.B.: $\Phi((x_1, x_2)) = (ax_1^2, bx_2^2, cx_1x_2, dx_1, ex_2, f)$

C und -C sind dann u.U. im m' -dimensionalen Raum linear separabel

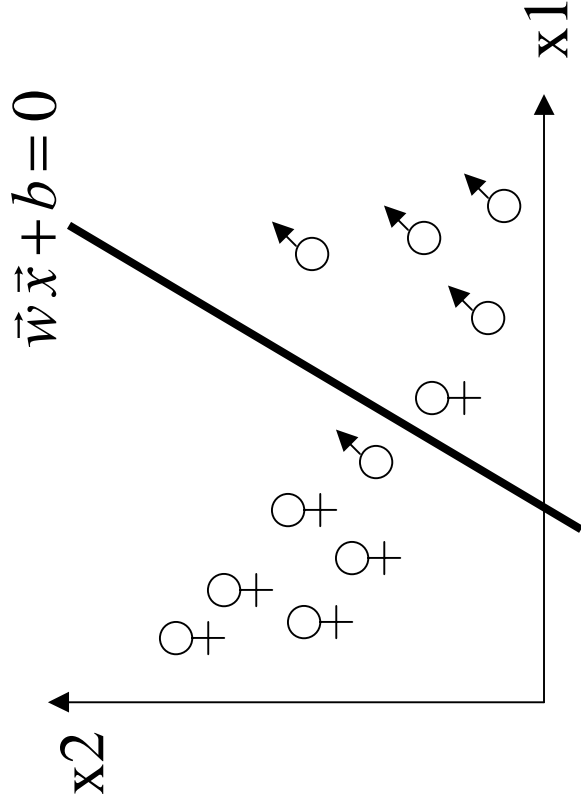
Für bestimmte Φ mit einer Kernfunktion $K(\vec{x}_i, \vec{x}_j) := \Phi(\vec{x}_i) \Phi(\vec{x}_j)$ bleiben Trainings- und Klassifikationsphase effizient,

z.B. für die Polynom-Familie $K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d$

→ Klassifikationstest für neuen Vektor \vec{y} : $b + \sum_{i=1}^n \alpha_i C_i K(\vec{x}_i, \vec{y}) > 0$

Erweiterte SVMs mit “weicher” Trennung

Falls Trainingsdaten nicht vollständig separabel, erlaube „Ausreißer“ auf der falschen Seite der Hyperebene



Finde $\vec{w} \in R^m$ und $b \in R$, so daß

1. $\|\vec{w}\| + \lambda \sum_{i=1}^n \varepsilon_i$ minimal ist und
2. $C_i \frac{1}{\|\vec{w}\|} (\vec{w} \vec{x}_i + b) \geq 1 - \varepsilon_i$

für alle $i=1, \dots, n$

mit Kontrollparameter λ zum Abwägen zwischen

Separationsweite $\delta = 1/\|\vec{w}\|$
und Fehlersumme $\sum_{i=1}^n \varepsilon_i$

9.2 Clustering: Klassifikation ohne Training (unsupervised classification)

gegeben:

n m -dimensionale Datensätze $d_j \in D$

$\subseteq \text{dom}(A_1) \times \dots \times \text{dom}(A_m)$ mit Attributen A_i

(z.B. Termhäufigkeitsvektoren $\subseteq N_0 \times \dots \times N_0$)

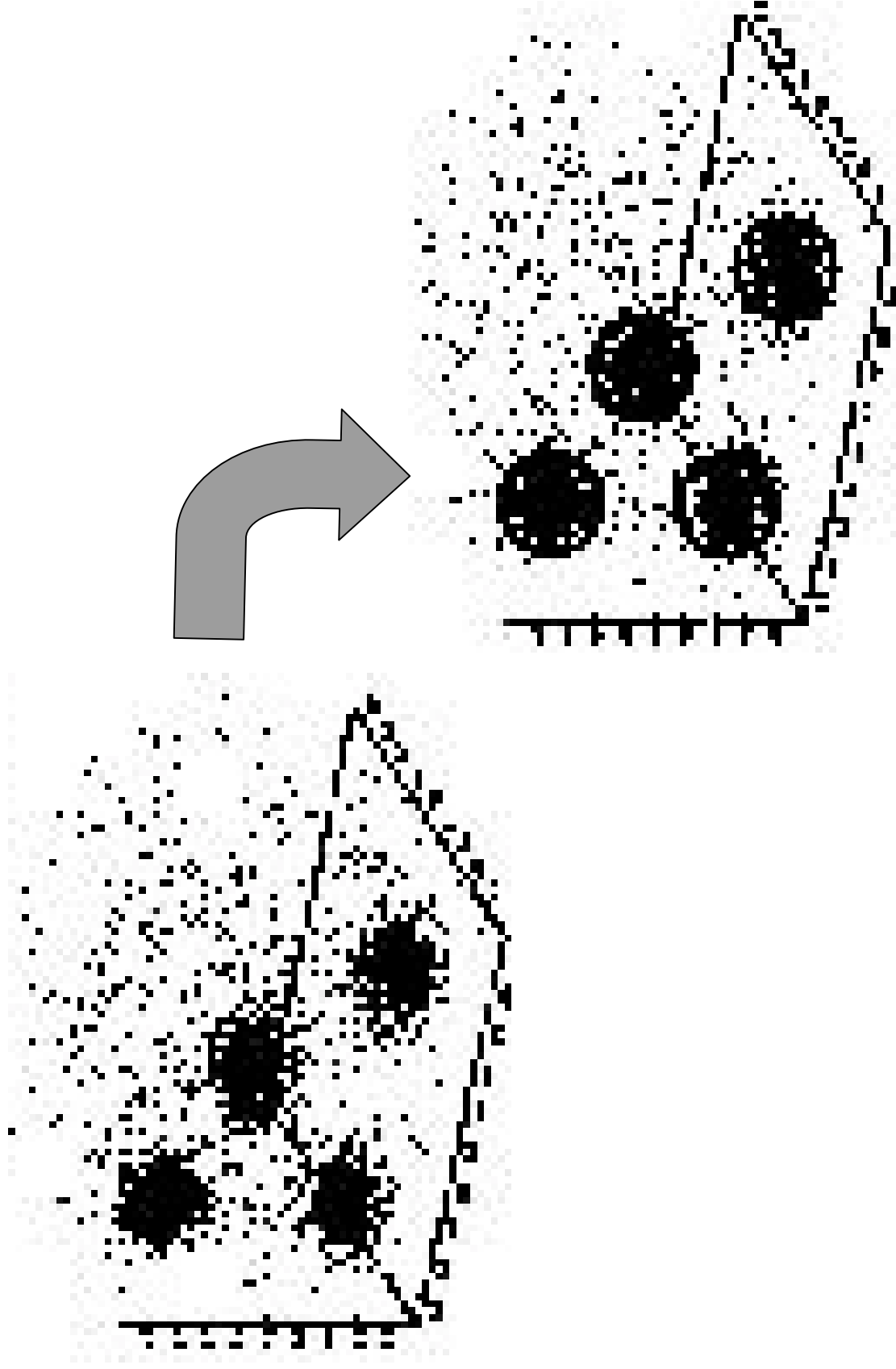
gesucht:

k **Cluster** c_1, \dots, c_k und eine Zuordnung $D \rightarrow \{c_1, \dots, c_k\}$, so daß die

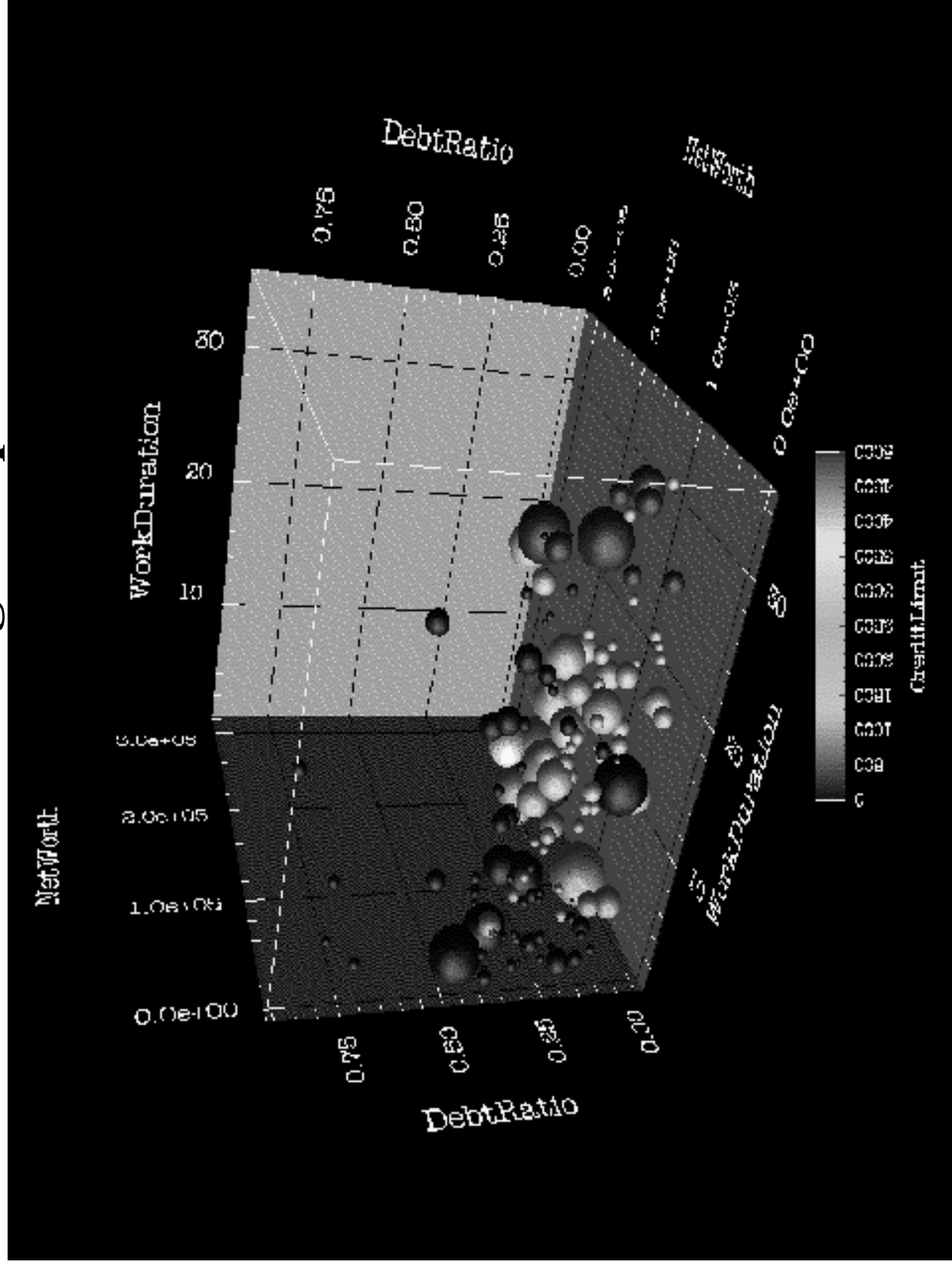
mittlere **Intra-Cluster-Ähnlichkeit** $\frac{1}{k} \sum_k \left(\frac{1}{|c_k|} \sum_{\vec{d} \in c_k} \text{sim}(\vec{d}, \vec{c}_k) \right)$
 möglichst groß und die
 mittlere **Inter-Cluster-Ähnlichkeit** $\frac{1}{k(k-1)} \sum_{i,j} \text{sim}(\vec{c}_i, \vec{c}_j)$ möglichst
 klein wird mit dem **Zentroid** („Gravitationszentrum“) \vec{c}_k von c_k :

$$\vec{c}_k = \frac{1}{|c_k|} \sum_{\vec{d} \in c_k} \vec{d}$$

Clustering-Beispiel 1



Clustering-Beispiel 2



Hierarchisches vs. Flaches Clustering

Hierarchisches Clustering:

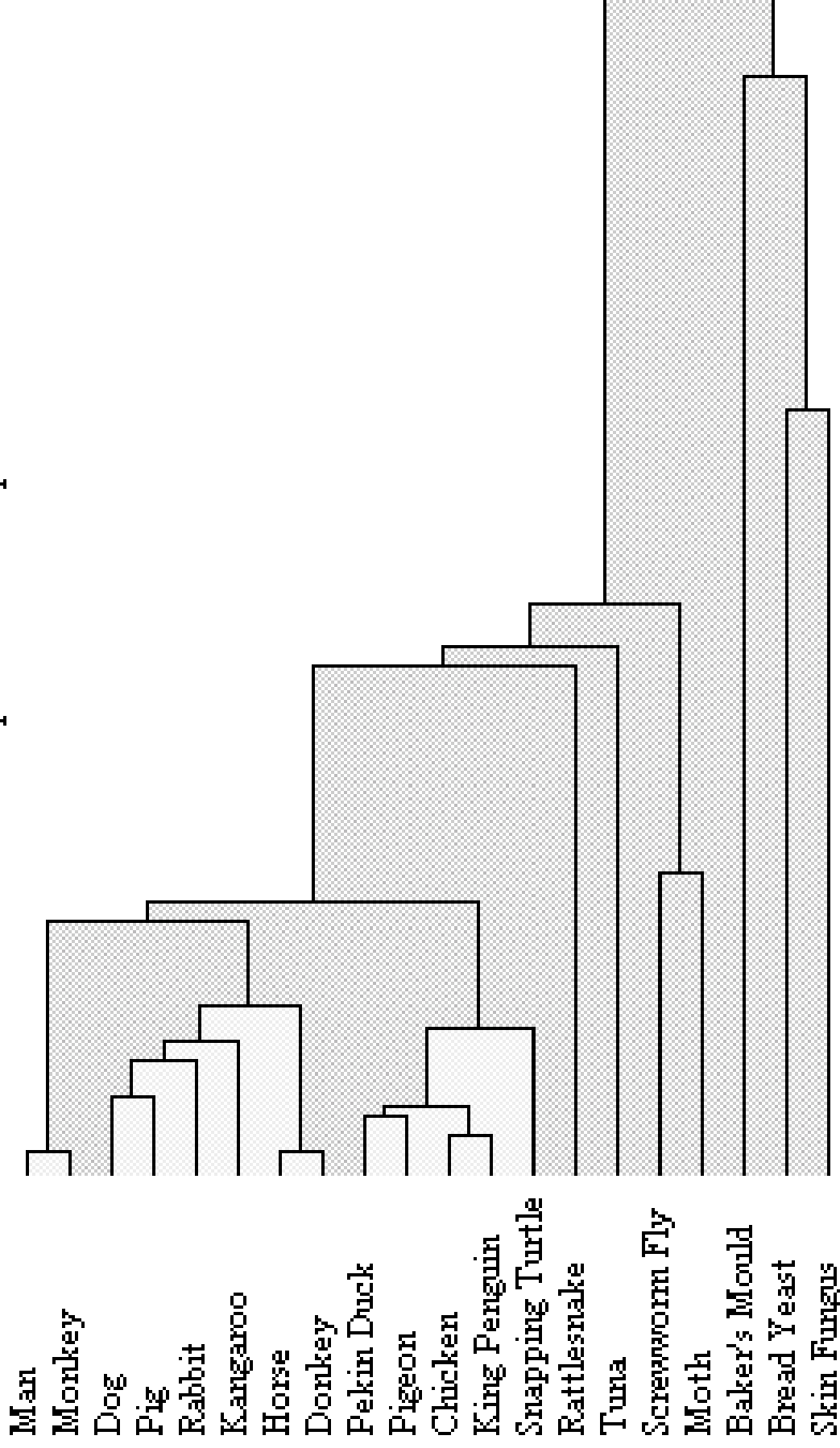
- detaillierter und u.U. aufschlußreicher
- Hierarchie entsteht in natürlicher Weise aus relativ simplen Algorithmen
- relativ aufwendig
- kein bevorzugter Algorithmus

Flaches Clustering:

- tendenziell gröbere Datenanalyse
- Detaillierungsgrad abhängig von der Wahl für die Anzahl der Cluster
- relativ effizient
- K-Means und EM sind simple Standardalgorithmen

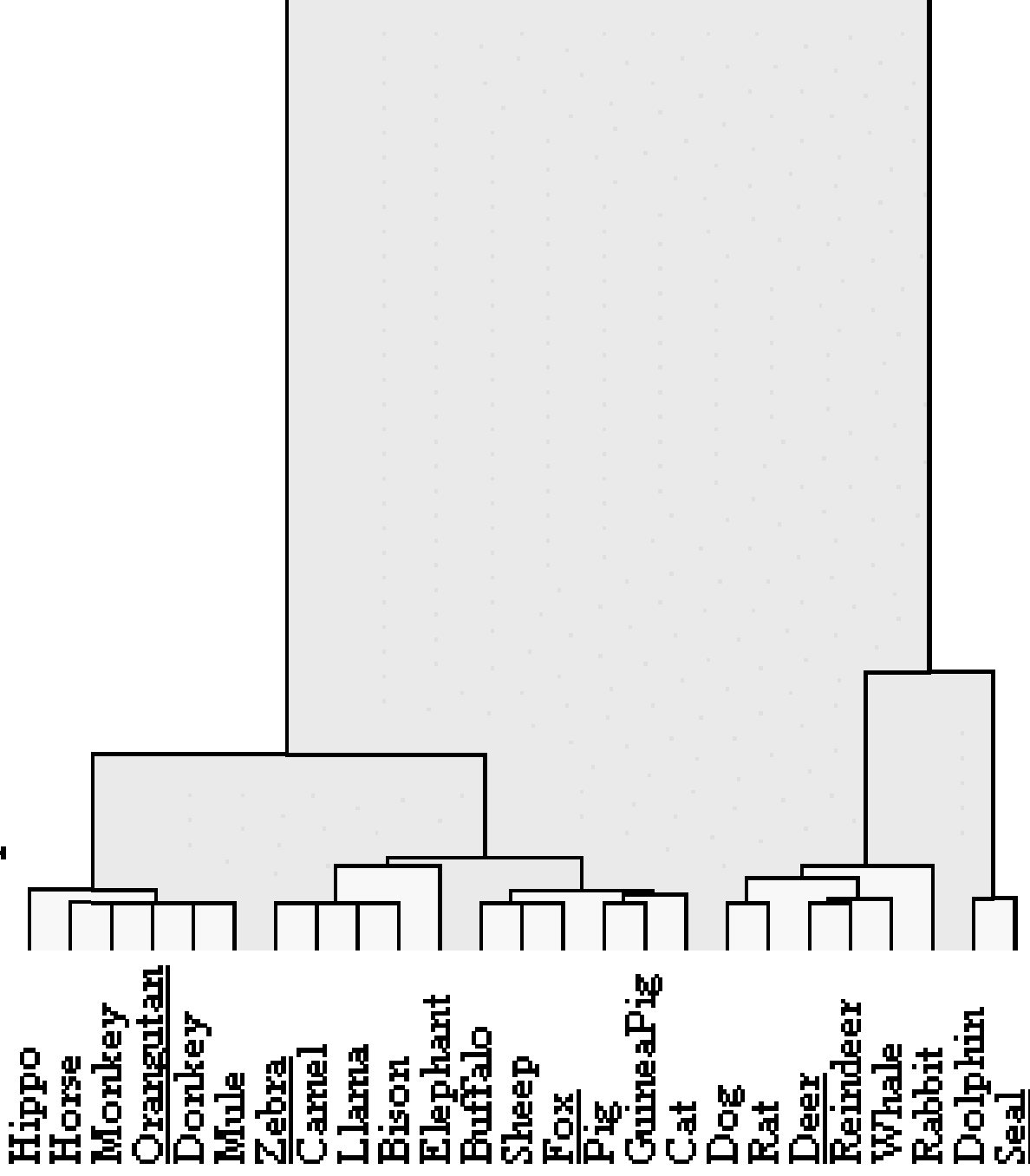
Beispiel für hierarchisches Clustering

Hierarchical classification of species based on proteins

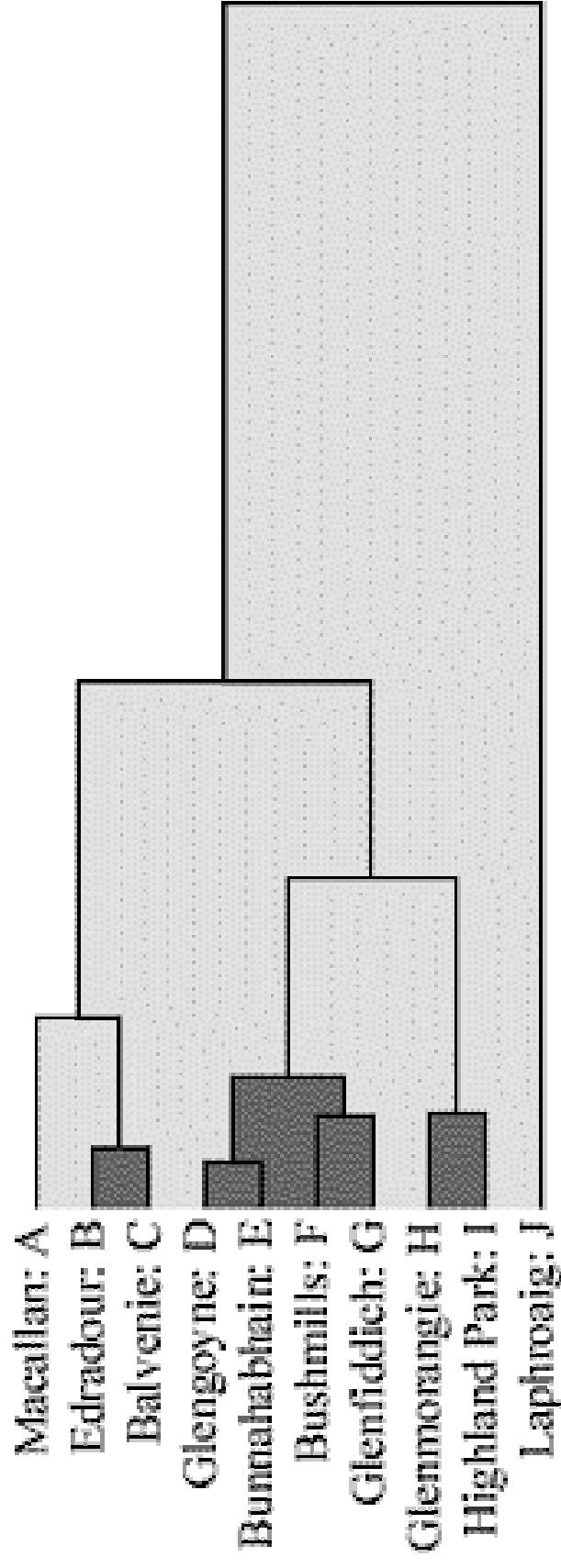


Beispiel 2 für hierarchisches Clustering

Composition of milk of 25 mammals



Beispiel 3 für hierarchisches Clustering



Agglomeratives Bottom-Up-Clustering

Prinzip:

- Beginne mit jedem Datensatz d_i als singulärem Cluster c_i
- Fasse pro Iteration die jeweils ähnlichsten Cluster c_i, c_j zu einem Cluster zusammen

```
for i:=1 to n do  $c_i := \{d_i\}$  od;  
 $C := \{c_1, \dots, c_n\}$ ; /* Menge aller Cluster */  
while  $|C| > 1$  do
```

```
    Bestimme  $c_i, c_j \in C$  mit maximaler Inter-Cluster-Ähnlichkeit;  
     $C := C - \{c_i, c_j\} \cup \{c_i \cup c_j\}$ ;  
od;
```

Divisives Top-Down-Clustering

Prinzip:

- Beginne mit einem einzigen Cluster, der alle Datensätze enthält
- Identifiziere pro Iteration den am wenigsten „kohärenten“ Cluster und teile ihn in zwei Cluster

```
c1 := {d1, ..., dn};
```

```
C := {c1}; /* Menge aller Cluster */
```

```
while es gibt ein  $c_j \in C$  mit  $|c_j| > 1$  do
```

 Bestimme c_i mit der kleinsten Intra-Cluster-Ähnlichkeit;

 Partitioniere c_i in c_i' und c_i'' (d.h. $c_i = c_i' \cup c_i''$ und $c_i' \cap c_i'' = \emptyset$)

 so daß die Inter-Cluster-Ähnlichkeit zwischen c_i' und c_i''

 minimal wird;

```
od;
```

Für die Partitionierung eines Clusters verwendet man selbst
Wiederum ein Clustering-Verfahren (z.B. ein Bottom-Up-Verfahren)

Alternative Ähnlichkeitsmetriken für Cluster

Gegeben: Ähnlichkeit auf Datensätzen - sim: $D \times D \rightarrow \mathbb{R}$ oder $[0,1]$

Definiere: Ähnlichkeit zwischen Clustern – sim: $2^D \times 2^D \rightarrow \mathbb{R}$ oder $[0,1]$

Alternativen:

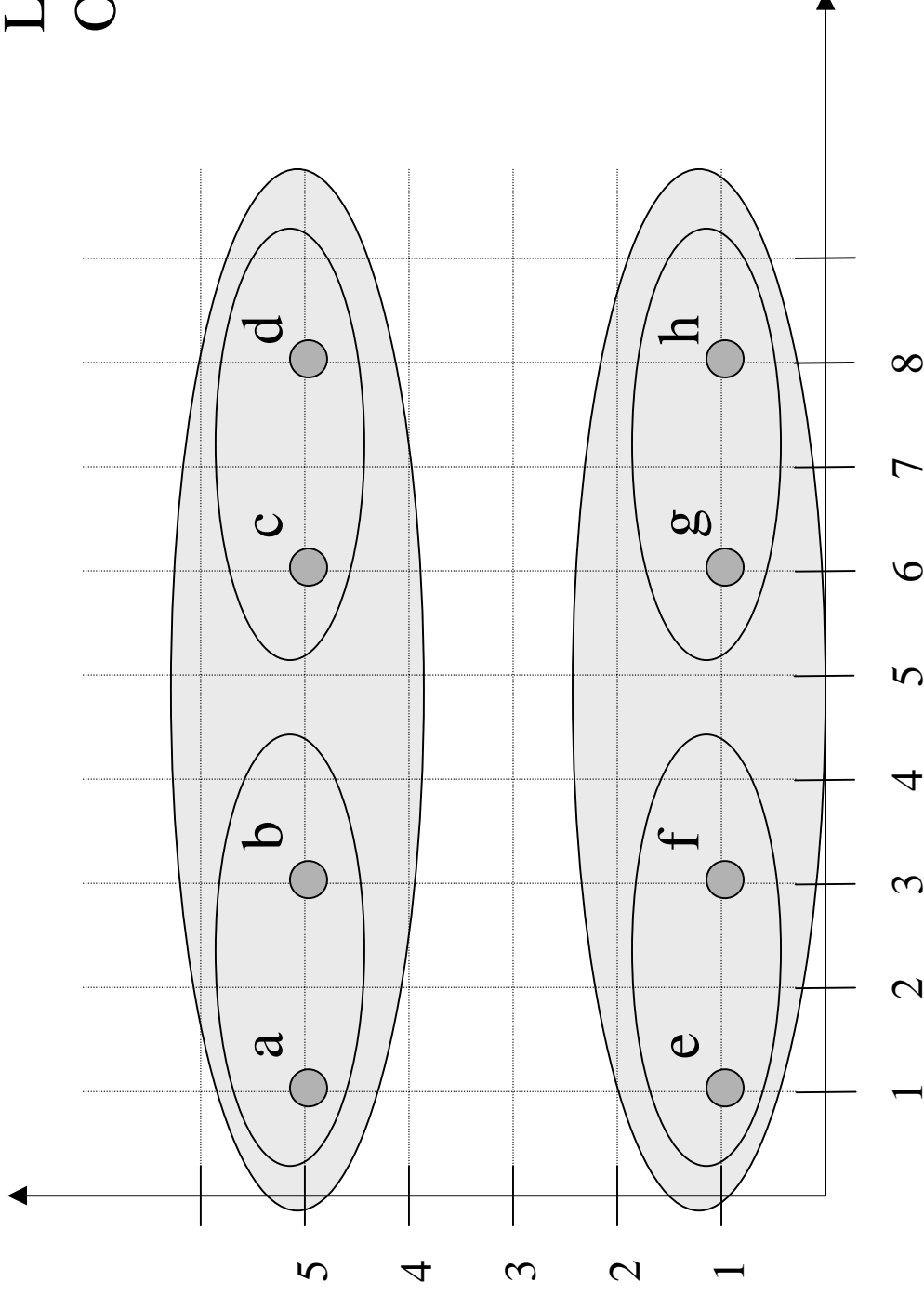
- **Zentroid-Methode:** $\text{sim}(c, c') = \text{sim}(d, d')$ mit Zentroid d von c und Zentroid d' von c'
- **Single-Link-Methode:** $\text{sim}(c, c') = \text{sim}(d, d')$ mit $d \in c, d' \in c'$, so daß d und d' die größte Ähnlichkeit haben
- **Complete-Link-Methode:** $\text{sim}(c, c') = \text{sim}(d, d')$ mit $d \in c, d' \in c'$, so daß d und d' die kleinste Ähnlichkeit haben
- **Group-Average-Methode:** $\frac{1}{|c| \cdot |c'|} \sum_{d \in c, d' \in c'} \text{sim}(d, d')$

Für hierarchisches Clustering muß gelten:

$$\max \{ \text{sim}(c, c'), \text{sim}(c, c'') \} \geq \text{sim}(c, c' \cup c'') \text{ für alle } c, c', c'' \in 2^D$$

Beispiel für Bottom-Up-Clustering mit der Single-Link-Metrik

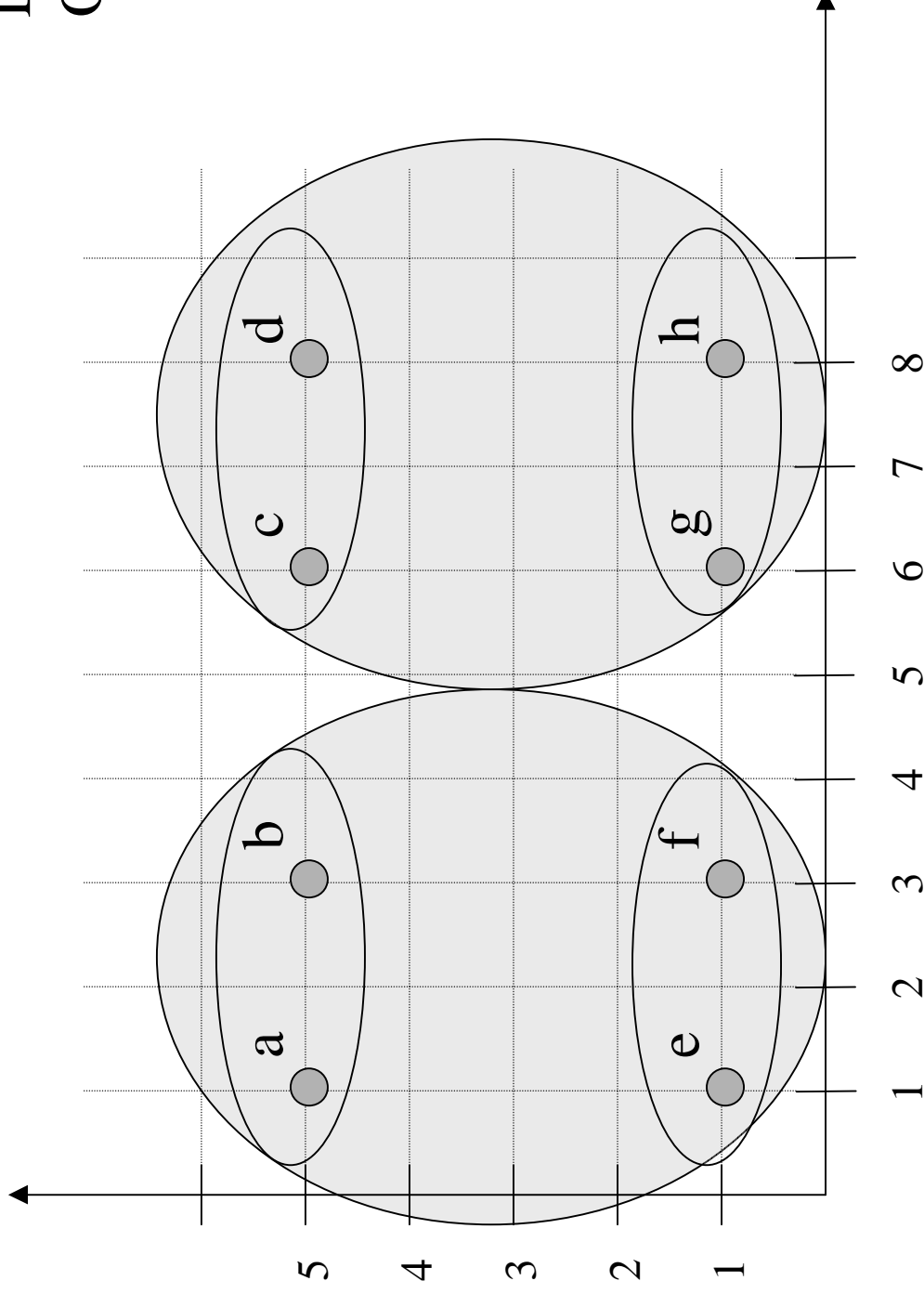
Laufzeit:
 $O(n^2)$



Betonung lokaler Cluster-Kohärenz → Tendenz zu länglichen Clustern

Beispiel für Bottom-Up-Clustering mit der Complete-Link-Metrik

Laufzeit:
 $O(n^3)$



Betonung globaler Cluster-Kohärenz → Tendenz zu runderen Clustern

Bezug zu Graphalgorithmen

Single-Link-Clustering:

- entspricht der Konstruktion eines maximalen Spannbaums für den gewichteten Graphen $G = (V, E)$ mit $V=D$, $E=D \times D$ und Kantengewicht $\text{sim}(d, d')$ für $(d, d') \in E$
- Aus dem maximalen Spannbaum kann die Cluster-Hierarchie durch rekursives Entfernen der kürzesten Kante konstruiert werden.

Single-Link-Clustering ist auch eng mit dem Problem verwandt, maximale Zusammenhangskomponenten in einem Graph zu finden. (und zwar auf einem Graph, der nur Kanten (d, d') enthält, so daß $\text{sim}(d, d')$ über einem Schwellwert liegt)

Complete-Link-Clustering ist eng mit dem Problem verwandt, maximale Cliques in einem Graph zu finden.

Bottom-Up-Clustering mit der Group-Average-Metrik (1)

Merge-Schritt vereinigt diejenigen beiden Cluster c_i und c_j ,
so daß die Intra-Cluster-Ähnlichkeit von $c := c_i \cup c_j$

$$S(c) := \frac{1}{|c| \cdot (|c| - 1)} \sum_{\substack{d, d' \in c \\ d \neq d'}} \text{sim}(d, d') \quad \text{maximal wird}$$

Naive Implementierung hat Laufzeit $O(n^3)$:
 $n-1$ Merge-Schritte mit jeweils $O(n^2)$ Berechnungsschritten

Bottom-Up-Clustering mit der Group-Average-Metrik (2)

Effiziente Implementierung – mit Gesamtlaufzeit $O(n^2)$ –
möglich für Cosinus-Ähnlichkeit mit normierten Vektoren
der Länge 1, also Skalarprodukt-Ähnlichkeit:

Einmalige Berechnung der Ähnlichkeit aller Dokumentpaare
Berechnung von $\vec{s}(c) := \sum_{d \in c} d$
für jeden Cluster nach einem Merge-Schritt

Dann gilt:

$$S(ci \cup cj) = \frac{(\vec{s}(ci) + \vec{s}(cj)) \cdot (\vec{s}(ci) + \vec{s}(cj)) - (|ci| + |cj|)}{(|ci| + |cj|)(|ci| + |cj| - 1)}$$

Jeder Merge-Schritt kommt also mit konstantem Rechenaufwand aus

Single-Pass-Verfahren für flaches Clustering

Gegeben: Datensätze d_1, \dots, d_n

Gesucht: max. k Cluster $C := \{c_1, \dots, c_k\}$

$C := \{ \{d_1\} \};$ /* willkürliche Wahl für den ersten Cluster */
for $i:=2$ to n do

 Bestimme denjenigen Cluster c_j aus C mit dem größten

 Wert von $\text{sim}(d_i, c_j)$ (z.B. $\text{sim}(d_i, \vec{c}_j)$ mit Zentroid \vec{c}_j);

 if $\text{sim}(d_i, c_j) \geq \text{Schwellwert}$

 then Ordne d_i dem Cluster c_j zu

 else if $|C| < k$

 then $C := C \cup \{ \{d_i\} \};$ /* eröffne neuen Cluster */

 else Ordne d_i dem Cluster c_j zu

 fi

fi

od

K-Means-Methode für flaches Clustering (1)

Idee:

- Bestimme k Prototypvektoren, je einen pro Cluster
- Weise Datensätze dem jeweils ähnlichsten Prototypvektor zu und berechne neue Prototypvektoren (durch Mittelung über die zugewiesenen Vektoren)
- Iteriere, bis Cluster hinreichend stabil

Wähle zufällig k Prototypvektoren $\vec{c}_1, \dots, \vec{c}_k$

while kein Iterationsabbruch do

 for $i:=1$ to n do

 Teile d_i demjenigen Cluster c_j zu, für das $\text{sim}(\vec{d}_i, \vec{c}_j)$ minimal ist
 od;

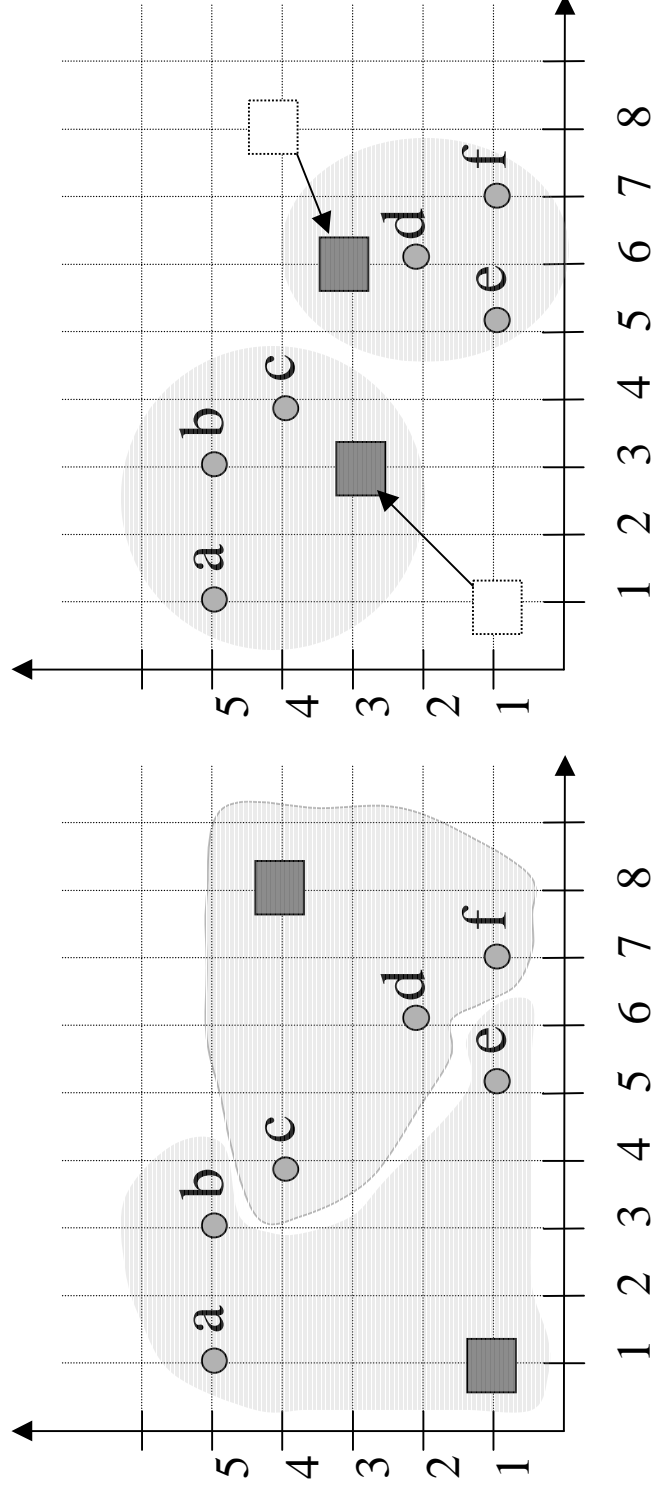
 for $j:=1$ to k do $\vec{c}_j := \frac{1}{|c_j|} \sum_{\vec{d} \in c_j} \vec{d}$ od;
od;

K-Means-Methode für flaches Clustering (2)

- Laufzeit ist $O(n)$ (bei konstanter Anzahl von Iterationen)
- Eine sinnvolle Anzahl k von Clustern könnte experimentell oder aufgrund des MDL-Prinzips bestimmt werden.
- Die initialen Prototypvektoren könnten auch mit einem anderen – effizienten – Clustering-Verfahren bestimmt werden
(z.B. Bottom-up-Clustering auf einer zufälligen Stichprobe der Datensätze).
- Für sim können beliebige Metriken verwendet werden.

Beispiel für K-Means-Clustering

K=2 ● Datensätze ■ Prototypvektoren



nach 1. Iteration

nach 2. Iteration

EM-Methode für flaches Clustering (Expectation Maximization)

Ansatz:

- Verallgemeinere K-Means-Verfahren, so daß jeder Datensatz bzgl. der k Cluster Zugehörigkeitswahrscheinlichkeiten hat gemäß einer parametrisierten (mehrdim.) Wahrscheinlichkeitsverteilung f
→ Zufallsvariable $Z_{ij} = 1$, falls d_i zu c_j gehört, 0 sonst
- Schätze die Parameter θ der postulierten W .verteilung $f(\theta, x)$, so daß die Wahrscheinlichkeit, daß die Datensätze eine Stichprobe dieser Verteilung sind, maximal wird
→ **Maximum-Likelihood-Schätzung:**
Maximiere $L(d_1, \dots, d_n, \theta) = P[d_1, \dots, d_n \text{ stammen aus } f(\theta, x)]$
bzw. maximiere $\log L$
falls analytisch zu schwer → wende **EM-Iterationsverfahren** an

Postulierte W .verteilung z.B.
Mixtur von k multivariaten Normalverteilungen

Simple Beispielfür Maximum-Likelihood-Schätzer

gegeben:

- Münze mit Bernoulli-Wahrscheinlichkeitsverteilung mit unbekanntem Parameter p für Zahl, $1-p$ für Kopf
 - Stichprobe (Daten): k -mal Zahl bei n Würfeln
- gesucht: Maximum-Likelihood-Schätzung für p

Sei $L(k, n, p) = P[\text{Stichprobe stammt aus Verteilung mit Param. } p]$

$$= \binom{n}{k} p^k (1-p)^{n-k}$$

Maximiere Log-Likelihood-Funktion $\log L(k, n, p)$:

$$\frac{\partial \log L}{\partial p} = \log \binom{n}{k} + k \log p + (n-k) \log (1-p) = 0$$

$$\Rightarrow p = \frac{k}{n}$$

Komplexeres Beispiel für Maximum-Likelihood-Schätzer

gegeben:

- Poisson-Verteilung mit Parameter λ (Erwartungswert)
- Stichprobe (Daten): Zahlen $x_1, \dots, x_n \in \mathbb{N}_0$

Sei r die größte unter diesen Zahlen,

und seien f_0, \dots, f_r die absoluten Häufigkeiten der Zahlen $0, \dots, r$.

gesucht: Maximum-Likelihood-Schätzung für λ

$$L(x_1, \dots, x_n, \lambda) = \prod_{i=0}^r \left(e^{-\lambda} \frac{\lambda^i}{i!} \right)^{f_i}$$

$$\Rightarrow \frac{\partial \ln L}{\partial \lambda} = \sum_{i=0}^r f_i \left(\frac{i}{\lambda} - 1 \right) = 0 \quad \Rightarrow \quad \hat{\lambda} = \frac{\sum_{i=0}^r i f_i}{\sum_{i=0}^r f_i} = \frac{1}{n} \sum_{i=1}^n x_i = \bar{x}$$

EM-Clustering-Methode mit Mixtur von k multivariaten Normalverteilungen

Annahme: Datensätze sind Stichprobe aus Mixtur von k multivariaten Normalverteilungen, also aus:

$$f(\vec{x}, \pi_1, \dots, \pi_k, \vec{\mu}_1, \dots, \vec{\mu}_k, \Sigma_1, \dots, \Sigma_k) = \sum_{j=1}^k \pi_j \frac{1}{\sqrt{(2\pi)^m |\Sigma_j|}} e^{-\frac{1}{2}(\vec{x} - \vec{\mu}_j)^T \Sigma_j^{-1} (\vec{x} - \vec{\mu}_j)}$$

mit Erwartungswerten $\vec{\mu}_j$
und invertierbaren, positiv definiten, symmetrischen
m×m-Covarianz-Matrizen Σ_j

→ Maximiere die Log-Likelihood-Funktion:

$$\log L(\vec{x}_1, \dots, \vec{x}_n, \theta) := \log \prod_{i=1}^n P[\vec{x}_i | \theta] = \sum_{i=1}^n \left(\log \sum_{j=1}^k \pi_j n(\vec{x}_i, \vec{\mu}_j, \Sigma_j) \right)$$

EM-Iterationsverfahren (1)

Initialisierung des EM-Verfahrens z.B. durch:
 $\pi_1 = \dots = \pi_k = 1/k$, Wahl der k-Means-Cluster-Zentroide für $\vec{\mu}_1, \dots, \vec{\mu}_k$
und einer Einheitsmatrix (1 auf der Diagonalen) für $\Sigma_1, \dots, \Sigma_k$

Iteriere, bis sich Parameterschätzungen nur noch wenig ändern:

1) Expectation-Schritt:

Berechne $E[Z_{ij}]$ auf der Basis der letzten Schätzung

für θ , d.h. π_1, \dots, π_k , $\vec{\mu}_1, \dots, \vec{\mu}_k$ und $\Sigma_1, \dots, \Sigma_k$

2) Minimization-Schritt:

Verbessere die Parameterschätzung für θ auf der Basis der letzten Werte von $E[Z_{ij}]$

Konvergenz ist garantiert, aber u.U. nur gegen lokales Maximum der Log-Likelihood-Funktion

EM-Iterationsverfahren (2)

Expectation-Schritt:

$$h_{ij} := E[Z_{ij} \mid \vec{x}_i, \theta] = \frac{P[\vec{x}_i \mid n_j(\theta)]}{\sum_{l=1}^k P[\vec{x}_i \mid n_l(\theta)]}$$

Maximization-Schritt:

$$\begin{aligned} \bar{\mu}_j &:= \frac{\sum_{i=1}^n h_{ij} \vec{x}_i}{\sum_{i=1}^n h_{ij}} & \Sigma_j &:= \frac{\sum_{i=1}^n h_{ij} (\vec{x}_i - \bar{\mu}_j)(\vec{x}_i - \bar{\mu}_j)^T}{\sum_{i=1}^n h_{ij}} \\ \pi_j &:= \frac{\sum_{i=1}^n h_{ij}}{k \frac{n}{\sum_{j=1}^k \sum_{i=1}^n h_{ij}}} = \frac{\sum_{i=1}^n h_{ij}}{n} \end{aligned}$$

EM-Clustering-Methode: Beispiel

Gegeben:

n=20 Terme aus Artikeln der New York Times:

ballot, polls, Gov, seats, profit, finance, payments, NFL, Reds,

Sox, inning, quarterback, score, scored, researchers, science,

Scott, Mary, Barbara, Edward

mit $m=20$ -dimensionalen Feature-Vektoren \vec{d}_i

mit $d_{ij} = \text{Anzahl der Artikel, in denen Term } i \text{ und Term } j \text{ auftreten}$

Ergebnis der EM-Iteration für die Schätzung der h_{ij} für $k=5$:

	1	2	3	4	5
ballot	0.63	0.12	0.04	0.09	0.11
polls	0.58	0.11	0.06	0.10	0.14
Gov	0.58	0.12	0.03	0.10	0.17
seats	0.55	0.14	0.08	0.08	0.15
profit	0.11	0.59	0.02	0.14	0.15
finance	0.15	0.55	0.01	0.13	0.16
payments	0.12	0.66	0.01	0.09	0.11
NFL	0.13	0.05	0.58	0.09	0.16
Reds	0.05	0.01	0.86	0.02	0.06
Sox	0.05	0.01	0.86	0.02	0.06

Clustering mit Dichteschätzer

Einflußfunktion (influence function) $g_y(x) : (R_0^+)^m \rightarrow R$

Einfluß eines Datensatzes y

auf einen Punkt x in seiner lokalen Umgebung

$$\text{z.B. } g_y(x) = e^{-\frac{\text{dist}(x,y)^2}{2\sigma^2}} \quad \text{mit } \text{dist}(x,y) := \frac{1}{1 + \text{sim}(x,y)}$$

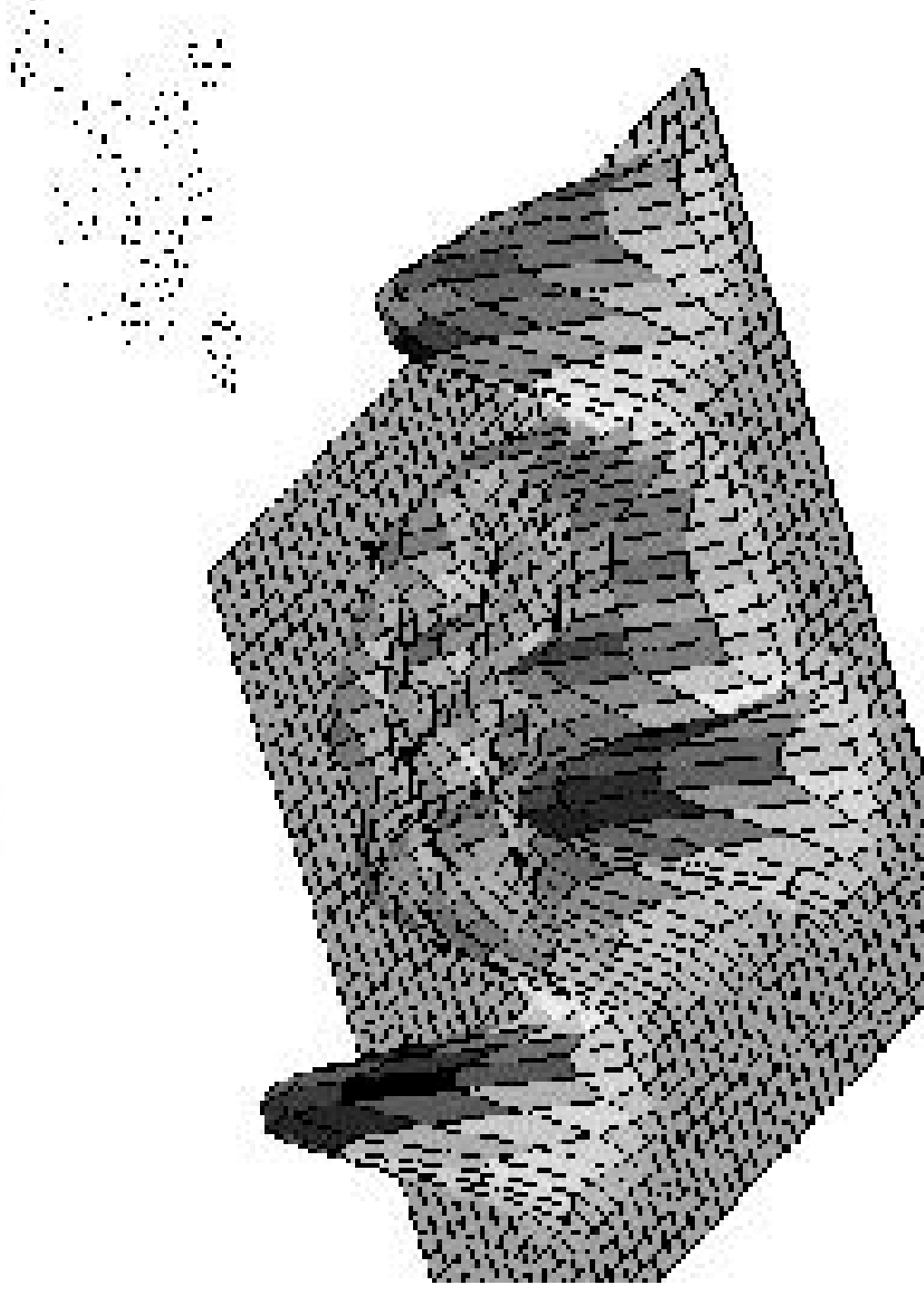
Dichtefunktion (density function) $f(x) : (R_0^+)^m \rightarrow R$

Dichte an Punkt x : Summe der Einflüsse aller y auf x

$$f(x) = \sum_{y \in D} g_y(x)$$

Cluster entsprechen lokalen Maxima der Dichtefunktion

Clustering mit Dichteschätzer: Beispiel



Quelle: D. Keim and A. Hinneburg, Clustering Techniques for Large Data Sets, Tutorial, KDD Conf. 1999

Automatische Bestimmung von Cluster-Labels

- Variante 1:

Klassifikation des Cluster-Zentroiden \vec{c}_k mit einem separaten supervised classifier

- Variante 2:

Wahl des Terms bzw. der Terme mit dem höchsten (tf*idf-) Gewicht im Cluster-Zentroiden \vec{c}_k

- Variante 2':

Berechnung eines approximativen Cluster-Zentroiden \vec{c}_k , aufgrund der m' ($m' \ll m$) wichtigsten Terme in den Dokumenten des Clusters und Wahl des wichtigsten Terms bzw. der wichtigsten Terme von \vec{c}_k

Cluster-basiertes Information Retrieval

Für eine Benutzer-Anfrage q:

- Bestimme Ranking der Cluster-Zentroiden bzgl. Q
- Initiiere Anfrage q auf dem Cluster bzw. den Clustern mit dem bzw. den ähnlichsten Zentroiden (ggf. mit Feedback des Benutzers)

Cluster-Browsing:

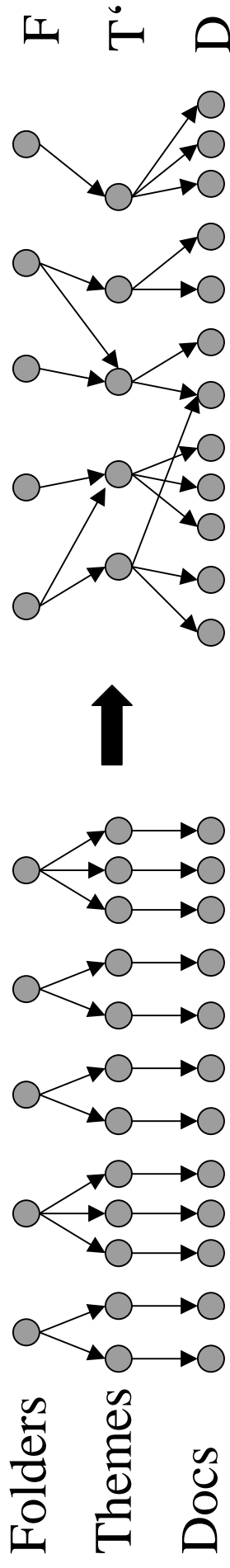
Benutzer kann durch Cluster-Hierarchie navigieren

Jeder Cluster ck ist durch seinen *Medoiden* repräsentiert:
das Dokument $d' \in ck$, für das die Summe $\sum_{d \in ck - \{d'\}} sim(d', d)$
maximal wird (oder die Ähnlichkeit zum Zentroiden)

Anwendung von Clustering auf Bookmark-Themen-Mining (1)

Gegeben: Benannte Bookmark-Folder einer Benutzer-Community

Gesucht: Themen des Benutzerinteressensprofil



Formalisierung:

Transformiere tripartiten Graph $G=(V,E)$ mit $V=F \cup T \cup D$ und

$E=\{(i,j) \in F \times T \mid \text{Folder } i \text{ enthält Dok. } j\} \cup \{(j,k) \in T \times D\}$

in bzgl. der Folder-Doc-Zuordnung möglichst „ähnlichen“

tripartiten Graph $G' = (V', E')$ mit $V'=F \cup T' \cup D$

mit möglichst geringen Kosten der Kantencodierung (MDL-Prinzip)

Anwendung von Clustering auf Bookmark-Themen-Mining (2): Codierungskosten

Ziel (*MDL-Prinzip*):

Minimiere Summe der Codierungskosten für

- 1) Kantencodierung von G' (*Mapping-Cost*) und
- 2) Kompensation der „Verzerrung“ von G' gegenüber G (*Distortion-Cost*)

Zu 1): basierend auf Entropie

$$H(G') = \sum_{v \in T'} \frac{\text{out degree}(v)}{|D|} \log_2 \frac{|D|}{\text{out degree}(v)} + \sum_{v \in T''} \frac{\text{out degree}(v)}{|F|} \log_2 \frac{|F|}{\text{out degree}(v)}$$

Zu 2): basierend auf Kullback-Leibler-Distanz (Cross Entropy: Divergenz von zwei Wahrscheinlichkeitsverteilungen) oder ähnlichen Maßen (z.B. Jensen-Shannon-Divergenz)

Anwendung von Clustering auf Bookmark-Themen-Mining (3): Heuristischer Algorithmus

Schritt 1: Konstruiere mittels agglomerativem Clustering einen Cluster-Baum auf D mit Dokumenten als Blättern

Schritt 2: Bestimme für jeden Folder $f \in F$ den bzgl. der Termverteilung „ähnlichsten“ Knoten des Cluster-Baums z.B. (mit binären Feature-Vektoren \vec{X}_i) denjenigen Knoten c mit der kleinsten Kullback-Leibler-Distanz:

$$KL(f || c) = \sum_{\vec{X} \in 2^m} P_f[\vec{X}] * \log_2 \frac{P_f[\vec{X}]}{P_c[\vec{X}]}$$

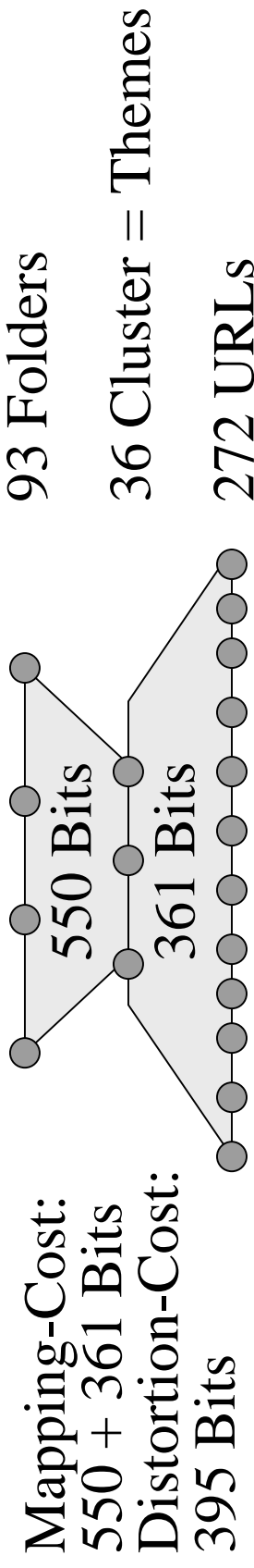
Schritt 3: Verbinde in G' f mit c und c mit allen Doks. in c

Schritt 4: Bestimme den bzgl. KL „verzerrtesten“ Folder f und Dokumente d aus f , die in G' nicht mehr mit f verbunden sind. Verbinde d mit $c \in T'$, wenn dadurch Mapping-Cost reduziert wird.

Anwendung von Clustering auf Bookmark-Themen-Mining (4): Experimente

Experiment mit 272 URLs aus 93 Folders
mit Namen „Linux“, „Radio“, „Books“ usw. und einigen 1000 Termen

Resultat:



nützlich u.a. zur Generierung von Cluster-Labels
mittels Bookmark-Namen, zum „Austausch“ von Bookmarks usw.

aus: S. Chakrabarti, Y. Batterywala.

Mining Themes from Bookmarks.

ACM SIGKDD Workshop on Text Mining, Boston, 2000.

9.3 Hypertext-Klassifikation

Idee: Berücksichtige Terme und ggf. Klassenzugehörigkeit
der via Hyperlink-Nachbardokumente
bei der Klassifikation eines Dokuments

Ansatz 1:

Erweitere jedes Dokumente um die Terme seiner Nachbarn
(ggf. nur im Hyperlink-Radius $\leq R$
oder mit Gewicht $\sim 1/r$ für Hyperlink-Distanz r)

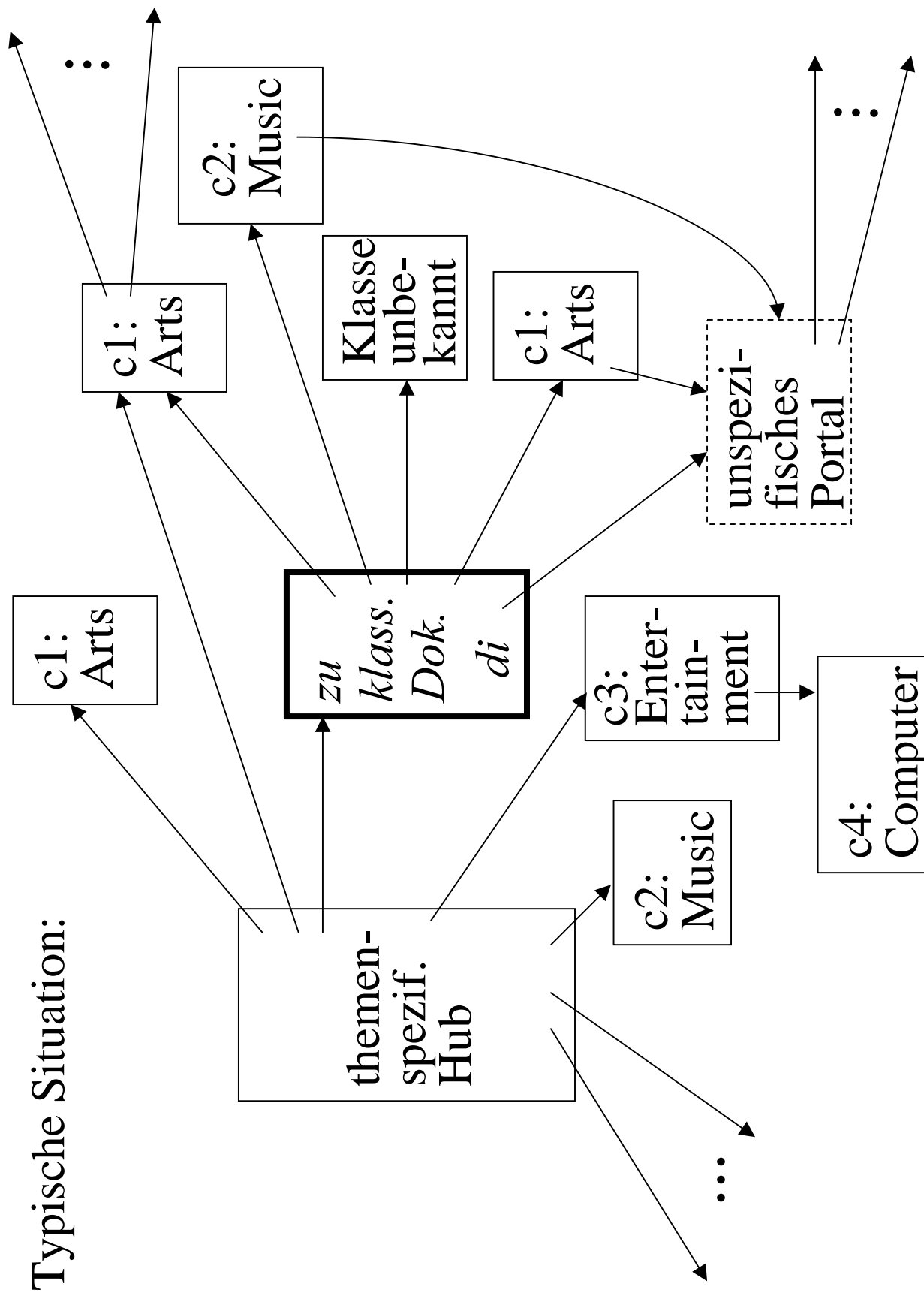
Problem: „Verrauschen“ des Dokumententhemas

Beispiel: Link von IR-Vorlesungs-Seite auf www.yahoo.com

Ansatz 2:

Berücksichtige bei einem zu klassifizierenden Dokument
die Klassenzugehörigkeit seiner Nachbarn

Berücksichtigung der Klassen von Nachbarn



Naive-Bayes-Klassifikator mit Berücksichtigung der Klassen von Nachbarn

Analysiere $P[d_i \in c_k | d_i \text{ hat } \vec{d}_i, \text{ Graph } G = (V, E) \text{ auf Korpus } D]$

bzw. $P[d_i \in c_k | d_i \text{ hat } \vec{d}_i \text{ und Nachbarn } N_i \text{ aus spezif. Klassen}]$

$$= \frac{P[\vec{d}_i, N_i | c_k] P[c_k]}{P[\vec{d}_i, N_i]} \sim P[\vec{d}_i, N_i | c_k] P[c_k]$$

$f(c_k, d_i, N_i) = P[\vec{d}_i, N_i | c_k] = P[\vec{d}_i | c_k] P[N_i | c_k]$ Unabhängigkeitsannahmen

$$= P[\vec{d}_i | c_k] \prod_{d_j \in \text{pred}(d_i)} P[d_j \in c(d_j) | d_i \in c_k] \prod_{d_j \in \text{succ}(d_i)} P[d_j \in c(d_j) | d_i \in c_k]$$

mit $\text{pred}(\text{di}) := \{dj \mid (j, i) \in E\}$ und $\text{succ}(\text{di}) := \{dj \mid (i, j) \in E\}$

$$= P[\vec{d}_i | c_k] * \prod_{v=1}^{\#Klassen} P[d_j \in c_v | d_i \in c_k, (j, i) \in E]^{|\text{pred}(d_i) \cap c_v|}$$

$$* \prod_{v=1}^{\#Klassen} P[d_j \in c_v | d_i \in c_k, (i, j) \in E]^{|\text{succ}(d_i) \cap c_v|}$$

Naive-Bayes-Klassifikator mit unbekannten Klassen von Nachbarn (1)

bei *bekannten* Klassen aller Nachbarn:

ordne d_i derjenigen Klasse c_k zu, für die $f(c_k, d_i, N_i)$ maximal ist

bei (z.T.) *unbekannten* Klassen der Nachbarn:

Iterationsverfahren (*Iterative Relaxation Labeling*):

Konstruiere Nachbarschaftsgraph $G_i = (N_i, E_i)$ mit Radius R um d_i
Klassifiziere alle Dokumente in N_i aufgrund von Textfeatures
Wiederhole bis hinreichende Konvergenz erreicht

Für alle d_j in N_i (inkl. d_i)

Berechne die Klassenzugehörigkeit von d_j aufgrund
der Textfeatures von d_j und der Klassen seiner Nachbarn
mittels Naive-Bayes

Naive-Bayes-Klassifikator mit unbekannten Klassen von Nachbarn (2)

Einteilung der Nachbardokumente in

N_i^K – Dok. mit bekannter Klasse und

N_i^U – Dok. mit a priori unbekannter Klasse.

Sei Δ^K die Gesamtheit der N_j^K für alle d_j im betrachteten Radius.

Dann ist:

$$P[d_i \in c_k \mid \Delta^K] = \sum_{N_i^U \in \Omega_i} \left(P[d_i \in c_k \mid N_i^U, \Delta^K] P[N_i^U \mid \Delta^K] \right)$$

mit der Menge Ω_i aller möglichen Klassenzuordnungen c von N_i^U

→ Iteration $P[d_i \in c_k \mid \Delta^K]^{(p+1)} :=$

$$\sum_{N_i^U \in \Omega_i} \left(P[d_i \in c_k \mid N_i^U, N_i^K] * \prod_{d_j \in N_i^U} \left(P[d_j \in c(d_j) \mid \Delta^K]^{(p)} \right) \right)$$

zur Konvergenz des Iterative Relaxation Labeling
siehe Theorie der Markov Random Fields

Experimentelle Resultate zur Hypertext-Klassifikation (1)

ca. 1000 Patente aus 12 Klassen:

Antenna, Modulator, Demodulator, Telephony, Transmission,
Motive, Regulator, Heating, Oscillator, Amplifier, Resistor, System.
Klassifikationsgenauigkeit mit reinen Textfeatures: 64%
Klassifikationsgenauigkeit mit Hyperlink-Klassifikation: 79%

ca. 20000 Web-Dokumente aus 13 Yahoo-Klassen:

Arts, Business&Economy, Computers&Internet, Education,
Entertainment, Government, Health, Recreation, Reference,
Regional, Science, Social Science, Society&Culture.
Klassifikationsgenauigkeit mit reinen Textfeatures: 32%
Klassifikationsgenauigkeit mit Hyperlink-Klassifikation: 79%

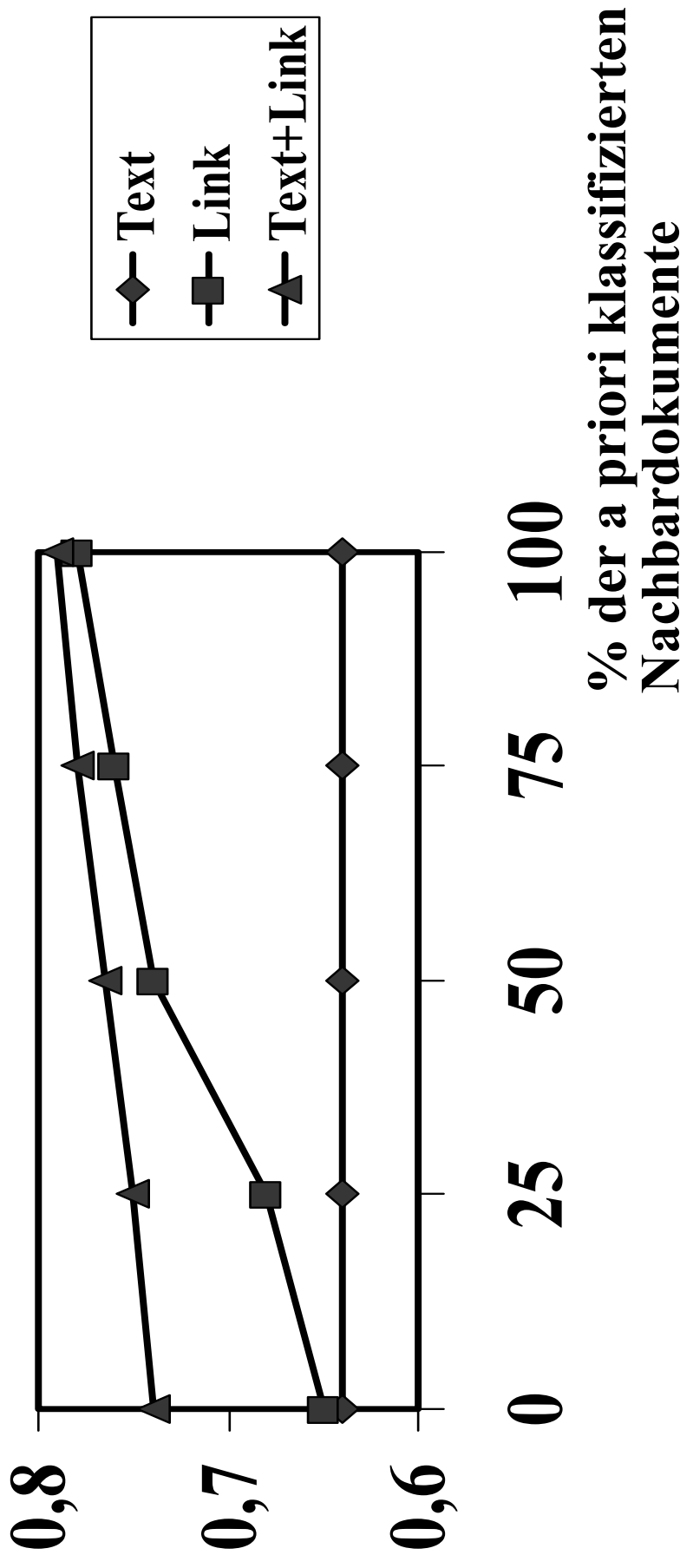
aus: S. Chakrabarti, B. Dom, P. Indyk.

Enhanced Hypertext Categorization Using Hyperlinks.

ACM SIGMOD International Conference on Management of Data, Seattle, 1998.

Experimentelle Resultate zur Hypertext-Klassifikation (2)

Genauigkeit



aus: S. Chakrabarti, B. Dom, P. Indyk.

Enhanced Hypertext Categorization Using Hyperlinks.

ACM SIGMOD International Conference on Management of Data, Seattle, 1998.

9.4 Hierarchische Textklassifikation mit Trainingsdaten

gegeben: Baum von Klassen mit Trainingsdokumenten

zu jedem Blatt oder jedem Knoten

gesucht: Zuordnung neuer Dokumente zu einer oder mehreren
Blättern oder Knoten

Top-Down-Ansatz 1 (zur Zuordnung zu genau einem Blatt):

Bestimme – von der Wurzel bis zu den Blättern –
auf jedem Baumniveau jeweils die Klasse,
zu der ein neues Dokument am besten passt.

Top-Down-Ansatz 2 (zur Zuordnung zu einem oder mehreren Knoten):

Bestimme – von der Wurzel bis zu den Blättern –
auf jedem Baumniveau alle Klassen, so daß die Wahrscheinlichkeit,
daß ein neues Dokument dazu passt, über einem Schwellwert liegt.

Kernproblem: Feature-Selektion

Bei der Entscheidung zwischen den Klassen einer Baumstufe ist – aus Effizienz- und vor allem auch Genauigkeitsgründen – eine Teilmenge geeigneter Features (Terme) auszuwählen. Diese Features sollen gute *Diskriminatoren* zwischen den Klassen derselben Baumstufe sein.

Beispiel:

Terme wie „Definition“, „Theorem“, „Lemma“ sind gute Diskriminatoren zwischen Arts, Entertainment, Science, etc. oder auch zwischen Biology, Mathematics, Social Sciences, etc.; sie sind schlechte Diskriminatoren zwischen den Unterklassen von Mathematics wie z.B. Algebra, Stochastics, etc.

Lösung: Betrachtung statistischer bzw. informationstheoretischer Maße zur Selektion geeigneter Features

Simple (klassenunabhängige) Kriterien zur Feature-Selektion

Häufigkeit (Document Frequency Thresholding):

Betrachte zum Test bzgl. Klasse c_j nur Terme t_i , die in mindestens df Trainingsdokumenten von c_j vorkommen.

Termstärke (Term Strength):

Zur Entscheidung zwischen Klassen c_1, \dots, c_k wähle diejenigen binären Features X_i (Termvorkommen) mit dem größten Wert von

$$s(X_i) := P[X_i \text{ kommt in Dok. } d \text{ vor} \mid X_i \text{ kommt in ähnl. Dok. } d' \text{ vor}]$$

Dabei wird die Menge der ähnlichen Dokumentpaare (d, d') durch einen Schwellwert bzgl. sim oder durch Clustering oder durch klassifizierte Trainingsdaten bestimmt.

+ weitere Kriterien dieser Bauart

Feature-Selektion auf der Basis des Informationsgewinns

Informationsgewinn (Information Gain):

Zur Entscheidung zwischen Klassen c_1, \dots, c_k wähle diejenigen binären Features X_i (Termvorkommen) mit dem größten Informationsgewinn

$$G(X_i) = \sum_{j=1}^k P[c_j] \log_2 \frac{1}{P[c_j]} \\ - P[X_i] \sum_{j=1}^k P[c_j | X_i] \log_2 \frac{1}{P[c_j | X_i]} \\ - P[\bar{X}_i] \sum_{j=1}^k P[c_j | \bar{X}_i] \log_2 \frac{1}{P[c_j | \bar{X}_i]}$$

Berechnung in Zeit $O(n) + O(mk)$
für n Trainingsdokumente, m Terme und k Klassen

Beispiel für Feature-Selektion auf der Basis des Informationsgewinns

	Film	Hit	Chart	Theorem	Limes	Integral	Gruppe	Vektor	
	f1	f2	f3	f4	f5	f6	f7	f8	
d1:	1	1	0	0	0	0	0	0	<div>Klassenbaum:</div> <div> <div> <div>Unterhaltung</div> <div>Mathematik</div> </div> <div> <div>Analysis</div> <div>Algebra</div> </div> </div>
d2:	0	1	1	0	0	0	1	0	
d3:	1	0	1	0	0	0	0	0	
d4:	0	1	1	0	0	0	0	0	
d5:	0	0	0	1	1	1	0	0	<div>Trainingsdokumente:</div> <div> <div>d1, d2, d3, d4</div> <div>→ Unterhaltung</div> <div>d5, d6, d7, d8</div> <div>→ Analysis</div> <div>d9, d10, d11, d12</div> <div>→ Algebra</div> </div>
d6:	0	0	0	1	0	1	0	0	
d7:	0	0	0	0	1	0	0	0	
d8:	0	0	0	1	0	1	0	0	
d9:	0	0	0	0	0	0	1	1	
d10:	0	0	0	1	0	0	1	1	
d11:	0	0	0	1	0	1	0	1	
d12:	0	0	1	1	1	0	1	0	

Beispielrechnung für Feature-Selektion auf der Basis des Informationsgewinns

Unterhaltung (d1-d4) vs. Mathematik (d5-d12):

$$G(\text{Film}) = 1/3 \log 3 + 2/3 \log 3/2 - 2/12 (1 \log 1 + 0 \log \infty) - 10/12 (2/10 \log 10/2 + 8/10 \log 10/8)$$

$$G(\text{Chart}) = 1/3 \log 3 + 2/3 \log 3/2 - 4/12 (3/4 \log 4/3 + 1/4 \log 4) - 8/12 (1/8 \log 8 + 7/8 \log 8/7)$$

$$G(\text{Theorem}) = 1/3 \log 3 + 2/3 \log 3/2 - 6/12 (0 \log \infty + 1 \log 1) - 6/12 (2/6 \log 3 + 4/6 \log 6/4)$$

$$G(\text{Vektor}) = 1/3 \log 3 + 2/3 \log 3/2 - 3/12 (0 \log \infty + 1 \log 1) - 9/12 (4/9 \log 9/4 + 5/9 \log 9/5)$$

⋮

Analysis (d5-d8) vs. Algebra (d9-d12):

$$G(\text{Film}) = 1/2 \log 2 + 1/2 \log 2 - 0 (1/2 \log 2 + 1/2 \log 2) - 1 (1/2 \log 2 + 1/2 \log 2)$$

$$G(\text{Theorem}) = 1/2 \log 2 + 1/2 \log 2 - 6/8 (1/2 \log 2 + 1/2 \log 2) - 2/8 (1/2 \log 2 + 1/2 \log 2)$$

$$G(\text{Integral}) = 1/2 \log 2 + 1/2 \log 2 - 4/8 (3/4 \log 4/3 + 1/4 \log 4) - 4/8 (1/4 \log 4 + 3/4 \log 4/3)$$

$$G(\text{Vektor}) = 1/2 \log 2 + 1/2 \log 2 - 3/8 (0 \log \infty + 1 \log 1) - 5/8 (4/5 \log 5/4 + 1/5 \log 5)$$

⋮

Feature-Selektion auf der Basis der Kullback-Leibler-Distanz

Kullback-Leibler-Distanz (Mutual Information, Cross Entropy):

Zur Entscheidung für Klasse c_j wähle diejenigen binären Features X_i (Termvorkommen) mit dem größten Wert von

$$MI(X_i, c_j) = P[X_i \wedge c_j] \log \frac{P[X_i \wedge c_j]}{P[X_i]P[c_j]}$$

oder

$$MI(X_i, c_j) = \sum_{X_i \in \{X_i, \bar{X}_i\}} \sum_{C \in \{c_j, \bar{c}_j\}} P[X \wedge C] \log \frac{P[X \wedge C]}{P[X]P[C]}$$

und für die Entscheidung bzgl. Klassen c_1, \dots, c_k :

$$MI(X_i) = \sum_{j=1}^k P[c_j] MI(X_i, c_j)$$

Berechnung in Zeit $O(n) + O(mk)$
für n Trainingsdokumente,
 m Terme und k Klassen

Beispielrechnung für Feature-Selektion auf der Basis der Kullback-Leibler-Distanz

Unterhaltung (d1-d4) vs. Mathematik (d5-d12):

$$\text{MI}(\text{Film}) = \frac{2}{12} \log \left[\frac{2}{12} / \left(\frac{2}{12} * \frac{1}{3} \right) \right] + 0 \log 0 + \frac{2}{12} \log \left[\frac{2}{12} / \left(\frac{2}{12} * \frac{1}{3} \right) \right] + \frac{8}{12} \log \left[\frac{8}{12} / \left(\frac{10}{12} * \frac{2}{3} \right) \right]$$

$$\text{MI}(\text{Chart}) = \frac{3}{12} \log \left[\frac{3}{12} / \left(\frac{4}{12} * \frac{1}{3} \right) \right] + \frac{1}{12} \log \left[\frac{1}{12} / \left(\frac{4}{12} * \frac{2}{3} \right) \right] + \frac{1}{12} \log \left[\frac{1}{12} / \left(\frac{8}{12} * \frac{1}{3} \right) \right] + \frac{7}{12} \log \left[\frac{7}{12} / \left(\frac{8}{12} * \frac{2}{3} \right) \right]$$

$$\text{MI}(\text{Theorem}) = 0 \log 0 + \frac{6}{12} \log \left[\frac{6}{12} / \left(\frac{6}{12} * \frac{2}{3} \right) \right] + \frac{4}{12} \log \left[\frac{4}{12} / \left(\frac{6}{12} * \frac{1}{3} \right) \right] + \frac{2}{12} \log \left[\frac{2}{12} / \left(\frac{6}{12} * \frac{2}{3} \right) \right]$$

Analysis (d5-d8) vs. Algebra (d9-d12):

$$\text{MI}(\text{Film}) = 0 \log 0 + 0 \log 0 + \frac{4}{8} \log \left[\frac{4}{8} / \left(\frac{8}{8} * \frac{1}{2} \right) \right] + \frac{4}{8} \log \left[\frac{4}{8} / \left(\frac{8}{8} * \frac{1}{2} \right) \right]$$

$$\text{MI}(\text{Theorem}) = \frac{3}{8} \log \left[\frac{3}{8} / \left(\frac{6}{8} * \frac{1}{2} \right) \right] + \frac{3}{8} \log \left[\frac{3}{8} / \left(\frac{6}{8} * \frac{1}{2} \right) \right] + \frac{1}{8} \log \left[\frac{1}{8} / \left(\frac{2}{8} * \frac{1}{2} \right) \right] + \frac{1}{8} \log \left[\frac{1}{8} / \left(\frac{2}{8} * \frac{1}{2} \right) \right]$$

$$\text{MI}(\text{Vektor}) = 0 \log 0 + \frac{3}{8} \log \left[\frac{3}{8} / \left(\frac{3}{8} * \frac{1}{2} \right) \right] + \frac{4}{8} \log \left[\frac{4}{8} / \left(\frac{5}{8} * \frac{1}{2} \right) \right] + \frac{1}{8} \log \left[\frac{1}{8} / \left(\frac{5}{8} * \frac{1}{2} \right) \right]$$

Experimentelle Resultate zur hierarchischen Textklassifikation (1)

ca. 400 000 Dokumente (von www.looksmart.com)
aus ca. 17000 Klassen in 7 Stufen:

13 Klassen auf Stufe 1 (Automotive, Business&Finance,
Computers&Internet, Entertainment&Media, Health&Fitness, etc.),
150 Klassen auf Stufe 2

ca. 50 000 zufällig gewählte Dokumente als Trainingsdaten;
für jede der 13+150 Klassen Auswahl der 1000 Terme mit
dem höchsten Mutual-Information-Wert $MI(X,C)$

Automatische Klassifikation von 10 000 Dokumenten mit SVM
(mit Kontrollparameter $\lambda=0.01$):
Top-down-Zuordnung eines Dokuments zu allen Klassen, für die
Zuordnung über einem bestimmten Schwellwert δ lag
(mit einer Wahl von δ so daß
auf Trainingsdaten maximiert wurde

$$F = \frac{2 * Pr\text{ ä z i s i o n} * Ausbeute}{(Pr\text{ ä z i s i o n} + Ausbeute)})$$

aus: S. Dumais, H. Chen. Hierarchical Classification of Web Content. ACM SIGIR
Conference on Research and Development in Information Retrieval, Athens, 2000

Experimentelle Resultate zur Hypertext-Klassifikation (2)

Durchschnittliche (micro-averaged) Klassifikationsgüte auf

Stufe 1 (13 Klassen): $F \approx 0.572$

Stufe 2 (150 Klassen): $F \approx 0.476$

Beste und schlechteste Klassen:

$F \approx 0.841$ Health & Fitness / Drugs & Medicine

$F \approx 0.797$ Home & Family / Real Estate

$F \approx 0.841$ Reference & Education / K-12 Education

$F \approx 0.841$ Sports & Recreation / Fishing

$F \approx 0.034$ Society & Politics / World Culture

$F \approx 0.088$ Home & Family / For Kids

$F \approx 0.122$ Computers & Internet / News & Magazines

$F \approx 0.131$ Computers & Internet / Internet & the Web

aus: S. Dumais, H. Chen. Hierarchical Classification of Web Content. ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, 2000