

5. Klassifikation

Inhalt dieses Kapitels

5.1 Einleitung

Das Klassifikationsproblem, Bewertung von Klassifikatoren

5.2 Bayes-Klassifikatoren

Optimaler Bayes-Klassifikator, Naiver Bayes-Klassifikator, Anwendungen

5.3 Nächste-Nachbarn-Klassifikatoren

Grundbegriffe, Parameterwahl, Anwendungen

5.4 Entscheidungsbaum-Klassifikatoren

Grundbegriffe, Splitstrategien, Overfitting, Pruning von Entscheidungsbäumen

5.5 Skalierung für große Datenbanken

SLIQ, SPRINT, RainForest

5.1 Einleitung

Das Klassifikationsproblem

- Gegeben: eine Menge O von Objekten des Formats (o_1, \dots, o_d) mit Attributen $A_i, 1 \leq i \leq d$, und Klassenzugehörigkeit $c_i, c_i \in C = \{c_1, \dots, c_k\}$
- Gesucht:
 - die Klassenzugehörigkeit für Objekte aus $D \setminus O$
 - ein Klassifikator $K : D \rightarrow C$
- Abgrenzung zum Clustering
 - Klassifikation: Klassen apriori bekannt
 - Clustering: Klassen werden erst gesucht
- Verwandtes Problem: *Vorhersage* (Prediction)
 - gesucht ist der Wert für ein numerisches Attribut
 - Methode z.B. Regression



5.1 Einleitung

Beispiel

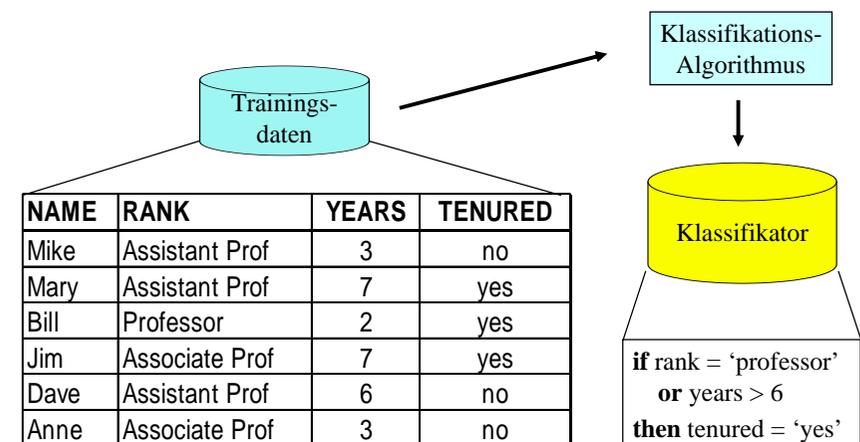
ID	Alter	Autotyp	Risiko
1	23	Familie	hoch
2	17	Sport	hoch
3	43	Sport	hoch
4	68	Familie	niedrig
5	32	LKW	niedrig

Einfacher Klassifikator

```
if Alter > 50 then Risikoklasse = Niedrig;
if Alter ≤ 50 and Autotyp=LKW then Risikoklasse=Niedrig;
if Alter ≤ 50 and Autotyp ≠ LKW
    then Risikoklasse = Hoch.
```

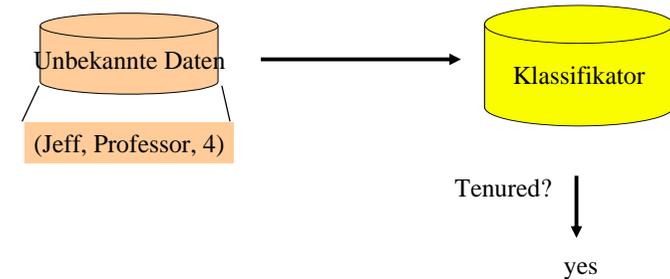
5.1 Der Prozess der Klassifikation

Konstruktion des Modells



5.1 Der Prozess der Klassifikation

Anwendung des Modells



 manchmal: keine Klassifikation unbekannter Daten sondern „nur“ besseres Verständnis der Daten

5.1 Bewertung von Klassifikatoren

Grundbegriffe

- Klassifikator ist für die Trainingsdaten optimiert, liefert auf der Grundgesamtheit der Daten evtl. schlechtere Ergebnisse
→ *Overfitting*
- *Train-and-Test*
Aufteilung der Menge O in zwei Teilmengen:
 - *Trainingsmenge*
zum Lernen des Klassifikators (Konstruktion des Modells)
 - *Testmenge*
zum Bewerten des Klassifikators

5.1 Bewertung von Klassifikatoren

Grundbegriffe

- Train-and-Test nicht anwendbar, wenn nur wenige Objekte mit bekannter Klassenzugehörigkeit
- Stattdessen: *m*-fache *Überkreuz-Validierung* (*Cross-Validation*)
- Idee
 - teile die Menge O in m gleich große Teilmengen
 - verwende jeweils $m-1$ Teilmengen zum Training und die verbleibende Teilmenge zur Bewertung
 - kombiniere die erhaltenen m Klassifikationsfehler (und die m gefundenen Modelle!)

5.1 Bewertung von Klassifikatoren

Gütemaße für Klassifikatoren

- Klassifikationsgenauigkeit
- Kompaktheit des Modells
z.B. Größe eines Entscheidungsbaums
- Interpretierbarkeit des Modells
wieviel Einsichten vermittelt das Modell dem Benutzer?
- Effizienz
der Konstruktion des Modells
der Anwendung des Modells
- Skalierbarkeit für große Datenmengen
für sekundärspeicherresidente Daten
- Robustheit
gegenüber Rauschen und fehlenden Werten

5.1 Bewertung von Klassifikatoren

Gütemaße für Klassifikatoren

- Sei K ein Klassifikator, $TR \subseteq O$ die Trainingsmenge, $TE \subseteq O$ die Testmenge. Bezeichne $C(o)$ die tatsächliche Klasse eines Objekts o .
- *Klassifikationsgenauigkeit* (classification accuracy) von K auf TE :

$$G_{TE}(K) = \frac{|\{o \in TE \mid K(o) = C(o)\}|}{|TE|}$$

- *Tatsächlicher Klassifikationsfehler* (true classification error)

$$F_{TE}(K) = \frac{|\{o \in TE \mid K(o) \neq C(o)\}|}{|TE|}$$

- *Beobachteter Klassifikationsfehler* (apparent classification error)

$$F_{TR}(K) = \frac{|\{o \in TR \mid K(o) \neq C(o)\}|}{|TR|}$$

5.2 Bayes-Klassifikatoren

Motivation

- gegeben ein Objekt o und zwei Klassen *positiv* und *negativ*
 - drei unabhängige Hypothesen h_1, h_2, h_3
 - die A-posteriori-Wahrscheinlichkeiten der Hypothesen für gegebenes o
 $P(h_1|o) = 0,4,$
 $P(h_2|o) = 0,3,$
 $P(h_3|o) = 0,3$
 - die A-posteriori-Wahrscheinlichkeiten der Klassen für gegebene Hypothese
 $P(\text{negativ}|h_1) = 0, P(\text{positiv}|h_1) = 1$
 $P(\text{negativ}|h_2) = 1, P(\text{positiv}|h_2) = 0$
 $P(\text{negativ}|h_3) = 1, P(\text{positiv}|h_3) = 0$
- ➡ o ist mit Wahrscheinlichkeit 0,4 *positiv*, mit Wahrscheinlichkeit 0,6 *negativ*

5.2 Optimaler Bayes-Klassifikator

Grundbegriffe

- Sei $H = \{h_1, \dots, h_n\}$ eine Menge *unabhängiger* Hypothesen.
- Sei o ein zu klassifizierendes Objekt.
- Der *optimale Bayes-Klassifikator* ordnet o der folgenden Klasse zu:

$$\operatorname{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|h_i) \cdot P(h_i|o)$$

- im obigen Beispiel: o als *negativ* klassifiziert
- Kein anderer Klassifikator mit demselben A-priori-Wissen erreicht im Durchschnitt eine bessere Klassifikationsgüte.

➡ *optimaler Bayes-Klassifikator*

5.2 Optimaler Bayes-Klassifikator

Grundbegriffe

- Vereinfachung: immer genau eine der Hypothesen h_i gültig
- vereinfachte Entscheidungsregel $\operatorname{argmax}_{c_j \in C} P(c_j|o)$



die $P(c_j|o)$ sind meist unbekannt

- Umformung mit Hilfe des Satzes von Bayes

$$\operatorname{argmax}_{c_j \in C} P(c_j|o) = \operatorname{argmax}_{c_j \in C} \frac{P(o|c_j) \cdot P(c_j)}{P(o)} = \operatorname{argmax}_{c_j \in C} P(o|c_j) \cdot P(c_j)$$

- endgültige Entscheidungsregel des optimalen Bayes-Klassifikators

$$\operatorname{argmax}_{c_j \in C} P(o|c_j) \cdot P(c_j)$$

➡ *Maximum-Likelihood-Klassifikator*

5.2 Naiver Bayes-Klassifikator

Grundbegriffe

- Schätzung der $P(c_j)$ als beobachtete Häufigkeit der einzelnen Klassen
- Schätzung der $P(o_i | c_j)$?
- Annahmen des naiven Bayes-Klassifikators
 - $o = (o_1, \dots, o_d)$
 - die Attributwerte o_i sind für eine gegebene Klasse *bedingt unabhängig*
- Entscheidungsregel des *naiven Bayes-Klassifikators*

$$\operatorname{argmax}_{c_j \in C} P(c_j) \cdot \prod_{i=1}^d P(o_i | c_j)$$

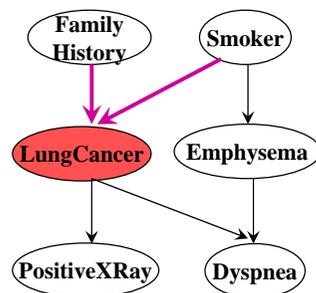
5.2 Bayes-Netzwerke

Grundbegriffe

- Graph mit Knoten = *Zufallsvariable* und Kante = *bedingte Abhängigkeit*
- Jede Zufallsvariable ist bei gegebenen Werten für die Vorgänger-Variablen bedingt unabhängig von allen Zufallsvariablen, die keine Nachfolger sind.
- Für jeden Knoten (Zufallsvariable): Tabelle der bedingten Wahrscheinlichkeiten
- Trainieren eines Bayes-Netzwerkes
 - bei gegebener Netzwerk-Struktur und allen bekannten Zufallsvariablen
 - bei gegebener Netzwerk-Struktur und teilweise unbekanntem Zufallsvariablen
 - bei apriori unbekannter Netzwerk-Struktur

5.2 Bayes-Netzwerke

Beispiel



	(FH,~S)	(~FH,~S)	(FH,S)	(~FH,S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

bedingte Wahrscheinlichkeiten für LungCancer

bei gegebenen Werten für FamilyHistory und Smoker liefert der Wert für Emphysema keine zusätzliche Information über LungCancer

5.2 Klassifikation von Texten

Grundlagen

- Anwendungen (z.B. [Craven et al. 1999], [Chakrabarti, Dom & Indyk 1998])
 - Filterung von Emails
 - Klassifikation von Webseiten
- Vokabular $T = \{t_1, \dots, t_d\}$ von relevanten Termen
- Repräsentation eines Textdokuments $o = (o_1, \dots, o_d)$
- o_i : Häufigkeit des Auftretens von t_i in o
- Methode
 - Auswahl der relevanten Terme
 - Berechnung der Termhäufigkeiten
 - Klassifikation neuer Dokumente

5.2 Klassifikation von Texten

Auswahl der Terme

- Reduktion der auftretenden Worte auf Grundformen

Stemming

Abhängigkeit von der Sprache der Texte

- Einwort- oder Mehrwort-Terme?
- Elimination von Stoppwörtern
- weitere Reduktion der Anzahl der Terme



bis zu 100 000 Terme

5.2 Klassifikation von Texten

Reduktion der Anzahl der Terme

- optimaler Ansatz

$O(2^{\text{AnzahlTerme}})$ Teilmengen

optimale Teilmenge läßt sich nicht effizient bestimmen

- Greedy-Ansatz

bewerte die „Separationsfähigkeit“ jedes Terms einzeln

sortiere die Terme nach dieser Maßzahl absteigend

wähle die ersten d Terme als Attribute aus

5.2 Klassifikation von Texten

Klassifikation neuer Dokumente

- Anwendung des naiven Bayes-Klassifikators



aber: Häufigkeiten der verschiedenen Terme typischerweise korreliert

- wichtigste Aufgabe:

Schätzung der $P(o_j | c)$ aus den Trainingsdokumenten

- Generierung eines Dokuments o der Klasse c mit n Termen

Bernoulli-Experiment:

n mal eine Münze werfen,

die für jeden Term t_i eine Seite besitzt

5.2 Klassifikation von Texten

Klassifikation neuer Dokumente

- Wahrscheinlichkeit, daß t_i nach oben kommt

$f(t_i, c)$: relative Häufigkeit des Terms t_i in der Klasse c

- Problem

– Term t_i tritt in keinem Trainingsdokument der Klasse c_j auf

– t_i tritt in einem zu klassifizierenden Dokument o auf

– in o treten ebenfalls „wichtige“ Terme der Klasse c_j auf



vermeide $P(o_j | c) = 0$

Glättung der absoluten Häufigkeiten

5.2 Klassifikation von Texten

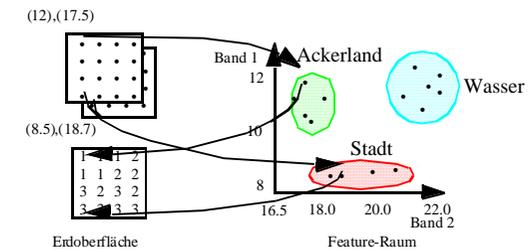
Experimentelle Untersuchung [Craven et al. 1999]

- Trainingsmenge: 4127 Webseiten von Informatik-Instituten
- Klassen: department, faculty, staff, student, research project, course, other
- 4-fache Überkreuz-Validierung
drei der Universitäten zum Training, vierte Universität zum Test
- Zusammenfassung der Ergebnisse
 - Klassifikationsgenauigkeit 70% bis 80 % für die meisten Klassen
 - Klassifikationsgenauigkeit 9% für Klasse *staff*
aber 80% korrekt in Oberklasse *person*
 - schlechte Klassifikationsgenauigkeit für Klasse *other*
große Varianz der Dokumente dieser Klasse

5.2 Interpretation von Rasterbildern

Motivation

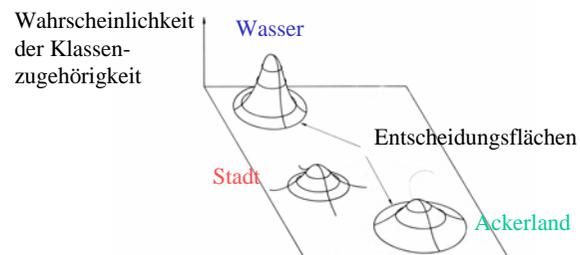
- *automatische* Interpretation von d Rasterbildern eines bestimmten Gebiets für jedes Pixel ein d -dimensionaler Grauwertvektor (o_1, \dots, o_d)
- verschiedene Oberflächenbeschaffenheiten der Erde besitzen jeweils ein charakteristisches Reflexions- und Emissionsverhalten



5.2 Interpretation von Rasterbildern

Grundlagen

- Anwendung des optimalen Bayes-Klassifikators
- Schätzung der $P(o|c)$ ohne Annahme der bedingten Unabhängigkeit
- Annahme einer d -dimensionalen Normalverteilung für die Grauwertvektoren einer Klasse



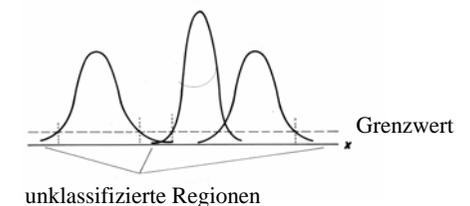
5.2 Interpretation von Rasterbildern

Methode

- Zu schätzen aus den Trainingsdaten
 - μ_i : d -dimensionaler Mittelwertvektor aller Feature-Vektoren der Klasse c_i
 - Σ_i : $d \cdot d$ Kovarianzmatrix der Klasse c_i

- Probleme der Entscheidungsregel

- Likelihood für die gewählte Klasse sehr klein
- Likelihood für mehrere Klassen ähnlich



5.2 Bayes-Klassifikatoren

Diskussion

- + Optimalitätseigenschaft
Standard zum Vergleich für andere Klassifikatoren
- + hohe Klassifikationsgenauigkeit in vielen Anwendungen
- + Inkrementalität
Klassifikator kann einfach an neue Trainingsobjekte adaptiert werden
- + Einbezug von Anwendungswissen
- Anwendbarkeit
die erforderlichen bedingten Wahrscheinlichkeiten sind oft unbekannt
- Ineffizienz
bei sehr vielen Attributen
insbesondere Bayes-Netzwerke

5.3 Nächste-Nachbarn-Klassifikatoren

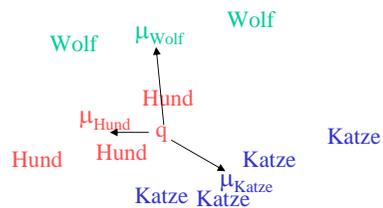
Motivation

- Bayes-Klassifikator zur Interpretation von Rasterbildern
Annahme der Normalverteilung der Grauwertvektoren einer Klasse
erfordert Schätzung von μ_i und Σ_i
Schätzung von μ_i benötigt wesentlich weniger Trainingsdaten
- Ziel
Klassifikator, der lediglich die Mittelwertvektoren für jede Klasse benötigt

➔ Nächste-Nachbarn-Klassifikatoren

5.3 Nächste-Nachbarn-Klassifikatoren

Beispiel



Klassifikator:
q ist ein Hund!

 Instance-Based Learning
verwandt dem Case-Based Reasoning

5.3 Nächste-Nachbarn-Klassifikatoren

Grundlagen

Basisverfahren

- Trainingsobjekte o als Attributvektoren $o = (o_1, \dots, o_d)$
- Bestimmung des Mittelwertvektors μ_i für jede Klasse c_i
- Zuordnung eines zu klassifizierenden Objektes zur Klasse c_i mit dem nächstgelegenen Mittelwertvektor μ_i

Verallgemeinerungen

- benutze mehr als ein Trainingsobjekt pro Klasse
- betrachte $k > 1$ Nachbarn
- gewichte die Klassen der k nächsten Nachbarn

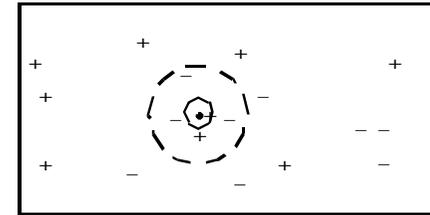
5.3 Nächste-Nachbarn-Klassifikatoren

Grundbegriffe

- Distanzfunktion
definiert die Ähnlichkeit (Unähnlichkeit) für Paare von Objekten
- Anzahl k der betrachteten Nachbarn
- *Entscheidungsmenge*
die Menge der zur Klassifikation betrachteten k -nächsten Nachbarn
- *Entscheidungsregel*
wie bestimmt man aus den Klassen der Entscheidungsmenge die Klasse des zu klassifizierenden Objekts?

5.3 Nächste-Nachbarn-Klassifikatoren

Beispiel



Klassen „+“ und „-“

- Entscheidungsmenge für $k = 1$
- Entscheidungsmenge für $k = 5$

Gleichgewichtung der Entscheidungsmenge

$k = 1$: Klassifikation als „+“, $k = 5$ Klassifikation als „-“

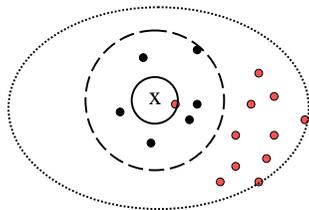
Gewichtung der Entscheidungsmenge nach inversem Quadrat der Distanz

$k = 1$ und $k = 5$: Klassifikation als „+“

5.3 Nächste-Nachbarn-Klassifikatoren

Wahl des Parameters k

- „zu kleines“ k : hohe Sensitivität gegenüber Ausreißern
- „zu großes“ k : viele Objekte aus anderen Clustern (Klassen) in der Entscheidungsmenge.
- mittleres k : höchste Klassifikationsgüte, oft $1 \ll k < 10$



x: zu klassifizieren

- Entscheidungsmenge für $k = 1$
- Entscheidungsmenge für $k = 7$
- Entscheidungsmenge für $k = 17$

5.3 Nächste-Nachbarn-Klassifikatoren

Entscheidungsregel

Standardregel

wähle die Mehrheitsklasse der Entscheidungsmenge

Gewichtete Entscheidungsregel

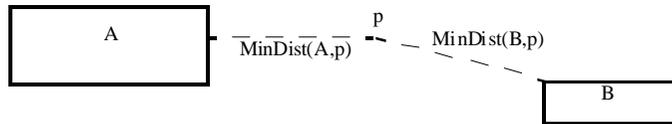
gewichte die Klassen der Entscheidungsmenge

- nach Distanz
- nach Verteilung der Klassen (oft sehr ungleich!)
Klasse A: 95 %, Klasse B 5 %
Entscheidungsmenge = {A, A, A, A, B, B, B}
Standardregel \Rightarrow A, gewichtete Regel \Rightarrow B

5.3 Nächste-Nachbarn-Klassifikatoren

Indexunterstützung für *k*-nächste-Nachbarn Anfragen

- balancierte Indexstruktur (z.B. X-Baum oder M-Baum)
- Anfragepunkt *p*
- PartitionList
 - MURs, deren referenzierte Teilbäume noch bearbeitet werden müssen, nach *MinDist* zu *p* aufsteigend sortiert
- NN
 - der nächste Nachbar von *p* in den bisher gelesenen Datenseiten



5.3 Nächste-Nachbarn-Klassifikatoren

Indexunterstützung für *k*-nächste-Nachbarn Anfragen

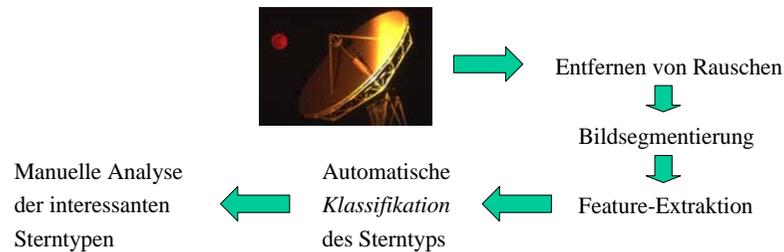
- alle MURs aus der PartitionList entfernen, die eine größere Distanz zum Anfragepunkt *p* besitzen als der bisher gefundene nächste Nachbar NN von *p*
- PartitionList wird aufsteigend nach *MinDist* zu *p* sortiert
- es wird jeweils das erste Element dieser Liste zur Bearbeitung ausgewählt
 - es werden keine überflüssigen Seiten gelesen!
- Anfragebearbeitung auf wenige Pfade der Indexstruktur beschränkt



Durchschnittliche Laufzeit $O(\log n)$ bei „nicht zu vielen“ Attributen
 bei sehr vielen Attributen $O(n)$

5.3 Klassifikation von Sternen

Analyse astronomischer Daten



Klassifikation des Sterntyps mit Nächste-Nachbarn-Klassifikator basierend auf dem Hipparcos-Katalog

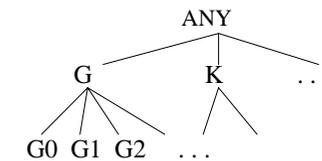
5.3 Klassifikation von Sternen

Hipparcos-Katalog [ESA 1998]

- enthält ca. 118 000 Sterne
- mit 78 Attributen (Helligkeit, Entfernung, Farbe, ...)
- Klassenattribut: Spektraltyp (Attribut H76)

z.B.

- H76: G0
- H76: G7.2
- H76: KIII/IV



- Werte des Spektraltyps sind vage
- Hierarchie von Klassen
 - ➔ benutze die erste Ebene der Klassenhierarchie

5.3 Klassifikation von Sternen

Verteilung der Klassen

Klasse	#Instanzen	Anteil Instanzen	
K	32 036	27.0	} häufige Klassen
F	25 607	21.7	
G	22 701	19.3	
A	18 704	15.8	
B	10 421	8.8	
M	4 862	4.1	} seltene Klassen
O	265	0.22	
C	165	0.14	
R	89	0.07	
W	75	0.06	
N	63	0.05	
S	25	0.02	
D	27	0.02	

5.3 Klassifikation von Sternen

Experimentelle Untersuchung [Poschenrieder 1998]

Distanzfunktion

- mit 6 Attributen (Farbe, Helligkeit und Entfernung)
 - mit 5 Attributen (ohne Entfernung)
- ⇒ beste Klassifikationsgenauigkeit mit 6 Attributen

Anzahl k der Nachbarn

⇒ beste Klassifikationsgenauigkeit für $k = 15$

Entscheidungsregel

- Gewichtung nach Distanz
 - Gewichtung nach Klassenverteilung
- ⇒ beste Klassifikationsgenauigkeit bei Gewichtung nach Distanz aber nicht nach Klassenverteilung

5.3 Klassifikation von Sternen

Klasse	Falsch klassifiziert	Korrekt klassifiziert	Klassifikationsgenauigkeit
K	408	2338	85.1%
F	350	2110	85.8%
G	784	1405	64.2%
A	312	975	75.8%
B	308	241	43.9%
M	88	349	79.9%
C	4	5	55.6%
R	5	0	0%
W	4	0	0%
O	9	0	0%
N	4	1	20%
D	3	0	0%
S	1	0	0%
Total	2461	7529	75.3%

 hohe Klassifikationsgenauigkeit für die häufigen Klassen, schlechte Genauigkeit für die seltenen Klassen
die meisten seltenen Klassen besitzen weniger als $k / 2 = 8$ Instanzen!

5.3 Nächste-Nachbarn-Klassifikatoren

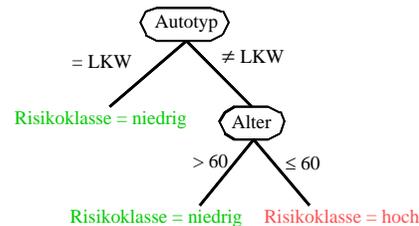
Diskussion

- + Anwendbarkeit
erfordert als Eingabe nur die Trainingsdaten
- + hohe Klassifikationsgenauigkeit
in vielen Anwendungen
- + Inkrementalität
Klassifikator kann sehr einfach an neue Trainingsobjekte adaptiert werden
- + auch zur Vorhersage einsetzbar
- Ineffizienz
erfordert k -nächste-Nachbarn Anfrage an die Datenbank
- liefert kein explizites Wissen über die Klassen

5.4 Entscheidungsbaum-Klassifikatoren

Motivation

ID	Alter	Autotyp	Risiko
1	23	Familie	hoch
2	17	Sport	hoch
3	43	Sport	hoch
4	68	Familie	niedrig
5	32	LKW	niedrig



finden explizites Wissen

Entscheidungsbäume sind für die meisten Benutzer verständlich

5.4 Entscheidungsbaum-Klassifikatoren

Grundbegriffe

- Ein *Entscheidungsbaum* ist ein Baum mit folgenden Eigenschaften:
 - ein innerer Knoten repräsentiert ein Attribut,
 - eine Kante repräsentiert einen Test auf dem Attribut des Vaterknotens,
 - ein Blatt repräsentiert eine der Klassen.
- Konstruktion eines Entscheidungsbaums
 - anhand der Trainingsmenge
 - Top-Down
- Anwendung eines Entscheidungsbaums
 - Durchlauf des Entscheidungsbaum von der Wurzel zu einem der Blätter
 - eindeutiger Pfad
 - Zuordnung des Objekts zur Klasse des erreichten Blatts

5.4 Entscheidungsbaum-Klassifikatoren

Konstruktion eines Entscheidungsbaums

Basis-Algorithmus

- Anfangs gehören alle Trainingsdatensätze zur Wurzel.
- Das nächste Attribut wird ausgewählt (Splitstrategie).
- Die Trainingsdatensätze werden unter Nutzung des Splitattributs partitioniert.
- Das Verfahren wird rekursiv für die Partitionen fortgesetzt.

→ lokal optimierender Algorithmus

Abbruchbedingungen

- keine weiteren Splitattribute
- alle Trainingsdatensätze eines Knotens gehören zur selben Klasse

5.4 Entscheidungsbaum-Klassifikatoren

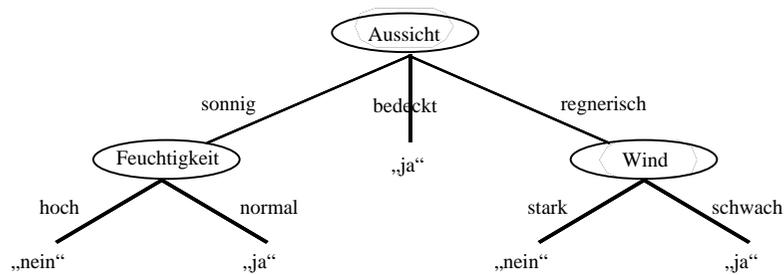
Beispiel

Tag	Aussicht	Temperatur	Feuchtigkeit	Wind	Tennispielen
1	sonnig	heiß	hoch	schwach	nein
2	sonnig	heiß	hoch	stark	nein
3	bedeckt	heiß	hoch	schwach	ja
4	regnerisch	mild	hoch	schwach	ja
5	regnerisch	kühl	normal	schwach	ja
6	regnerisch	kühl	normal	stark	nein
7

Ist heute ein Tag zum Tennispielen?

5.4 Entscheidungsbaum-Klassifikatoren

Beispiel

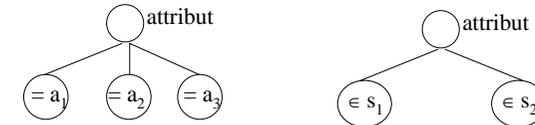


5.4 Splitstrategien

Typen von Splits

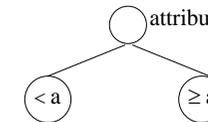
Kategorische Attribute

- Splitbedingungen der Form „attribut = a“ or „attribut ∈ set“
- viele mögliche Teilmengen



Numerische Attribute

- Splitbedingungen der Form „attribut < a“
- viele mögliche Splitpunkte



5.4 Splitstrategien

Qualitätsmaße für Splits

Gegeben

- eine Menge T von Trainingsobjekten
- eine disjunkte, vollständige Partitionierung T_1, T_2, \dots, T_m von T
- p_i die relative Häufigkeit der Klasse c_i in T

Gesucht

- ein Maß der *Unreinheit* einer Menge S von Trainingsobjekten in Bezug auf die Klassenzugehörigkeit
- ein Split von T in T_1, T_2, \dots, T_m , der dieses Maß der Unreinheit *minimiert*

➡ Informationsgewinn, Gini-Index

5.4 Splitstrategien

Informationsgewinn

- Entropie: minimale Anzahl von Bits zum Codieren der Nachricht, mit der man die Klasse eines zufälligen Trainingsobjekts mitteilen möchte
- Die *Entropie* für eine Menge T von Trainingsobjekten ist definiert als

$$\text{entropie}(T) = \sum_{i=1}^k p_i \cdot \log p_i$$

$$\text{entropie}(T) = 0, \text{ falls } p_i = 1 \text{ für ein } i$$

$$\text{entropie}(T) = 1 \text{ für } k = 2 \text{ Klassen mit } p_i = 1/2$$

- Das Attribut A habe die Partitionierung T_1, T_2, \dots, T_m erzeugt.
- Der *Informationsgewinn* des Attributs A in Bezug auf T ist definiert als

$$\text{informationsgewinn}(T, A) = \text{entropie}(T) - \sum_{i=1}^m \frac{|T_i|}{|T|} \cdot \text{entropie}(T_i)$$

5.4 Splitstrategien

Gini-Index

- *Gini-Index* für eine Menge T von Trainingsobjekten

$$gini(T) = 1 - \sum_{j=1}^k p_j^2$$

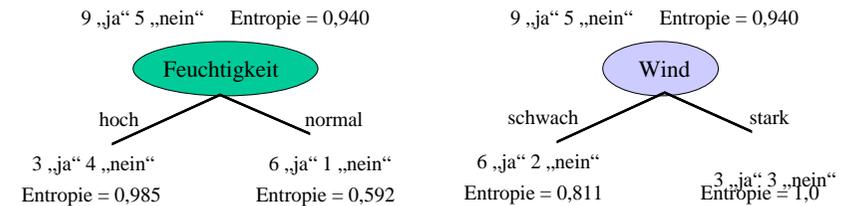
kleiner Gini-Index \Leftrightarrow geringe Unreinheit,
großer Gini-Index \Leftrightarrow hohe Unreinheit

- Das Attribut A habe die Partitionierung T_1, T_2, \dots, T_m erzeugt.
- *Gini-Index* des Attributs A in Bezug auf T ist definiert als

$$gini_A(T) = \sum_{i=1}^m \frac{|T_i|}{|T|} \cdot gini(T_i)$$

5.4 Splitstrategien

Beispiel



$$\text{informationsgewinn}(T, \text{Feuchtigkeit}) = 0,94 - \frac{7}{14} \cdot 0,985 - \frac{7}{14} \cdot 0,592 = 0,151$$

$$\text{informationsgewinn}(T, \text{Wind}) = 0,94 - \frac{8}{14} \cdot 0,811 - \frac{6}{14} \cdot 1,0 = 0,048$$

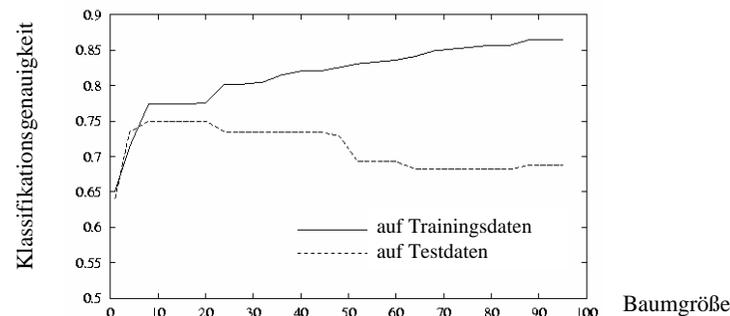
➡ Feuchtigkeit liefert den höheren Informationsgewinn

5.4 Overfitting

Einführung

Overfitting bei der Konstruktion eines Entscheidungsbaums, wenn es zwei Entscheidungsbäume E und E' gibt mit

- E hat auf der Trainingsmenge eine kleinere Fehlerrate als E' ,
- E' hat auf der Grundgesamtheit der Daten eine kleinere Fehlerrate als E .



5.4 Overfitting

Ansätze zum Vermeiden von Overfitting

- Entfernen von fehlerhaften Trainingsdaten
insbesondere widersprüchliche Trainingsdaten
- Wahl einer geeigneten Größe der Trainingsmenge
nicht zu klein, nicht zu groß
- Wahl einer geeigneten Größe des minimum support

minimum support:

Anzahl der Datensätze, die mindestens zu einem Blattknoten des Baums gehören müssen

 *minimum support* $\gg 1$

5.4 Overfitting

Ansätze zum Vermeiden von Overfitting

- Wahl einer geeigneten Größe der minimum confidence

minimum confidence: Anteil, den die Mehrheitsklasse eines Blattknotens mindestens besitzen muß



minimum confidence \ll 100%

Blätter können auch fehlerhafte Datensätze oder Rauschen „absorbieren“

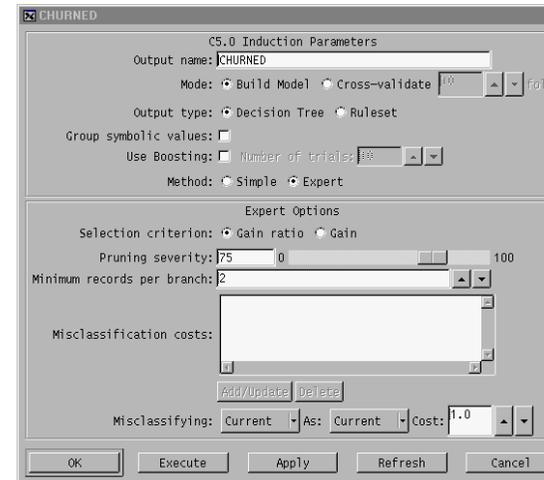
- nachträgliches Pruning des Entscheidungsbaums

Abschneiden der überspezialisierten Äste



nächster Abschnitt

5.4 Overfitting



Typische Parameter der Konstruktion eines Entscheidungsbaums (Clementine)

5.4 Pruning von Entscheidungsbäumen

Fehlerreduktions-Pruning [Mitchell 1997]

- Aufteilung der klassifizierten Daten in Trainingsmenge und Testmenge
- Konstruktion eines Entscheidungsbaums E für die Trainingsmenge
- Pruning von E mit Hilfe der Testmenge T
 - bestimme denjenigen Teilbaum von E , dessen Abschneiden den Klassifikationsfehler auf T am stärksten reduziert
 - entfernte diesen Teilbaum
 - fertig, falls kein solcher Teilbaum mehr existiert



nur anwendbar, wenn genügend viele klassifizierte Daten

5.4 Pruning von Entscheidungsbäumen

Minimales Kostenkomplexitäts-Pruning

[Breiman, Friedman, Olshen & Stone 1984]

- benötigt keine separate Testmenge
 - auch anwendbar für kleine Trainingsmengen
- Pruning des Entscheidungsbaums mit Hilfe der Trainingsmenge
 - Klassifikationsfehler ungeeignet als Qualitätskriterium
- Neues Qualitätskriterium für Entscheidungsbäume
 - Trade-off von Klassifikationsfehler und Baumgröße
 - gewichtete Summe aus Klassifikationsfehler und Baumgröße



kleinere Entscheidungsbäume generalisieren typischerweise besser

5.4 Pruning von Entscheidungsbäumen

Grundbegriffe

- Größe $|E|$ des Entscheidungsbaums E : Anzahl seiner Blätter
- *Kostenkomplexität* von E in Bezug auf die Trainingsmenge T und den Komplexitätsparameter $\alpha \geq 0$:

$$KK_T(E, \alpha) = F_T(E) + \alpha \cdot |E|$$

- Der *kleinste minimierende Teilbaum* $E(\alpha)$ von E in Bezug auf α erfüllt:
 - (1) es gibt keinen Teilbaum von E mit kleinerer Kostenkomplexität
 - (2) falls $E(\alpha)$ und B die Bedingung (1) erfüllen, ist $E(\alpha)$ Teilbaum von B
- $\alpha = 0$: $E(\alpha) = E$
- $\alpha = \infty$: $E(\alpha) =$ Wurzel von E
- $0 < \alpha < \infty$: $E(\alpha) =$ ein echter Teilbaum von E , mehr als die Wurzel

5.4 Pruning von Entscheidungsbäumen

Grundbegriffe

- E_e : Teilbaum mit der Wurzel e ,
 $\{e\}$: Baum, der nur aus dem Knoten e besteht.
- Für kleine Werte von α gilt: $KK_T(E_e, \alpha) < KK_T(\{e\}, \alpha)$,
 für große Werte von α gilt: $KK_T(E_e, \alpha) > KK_T(\{e\}, \alpha)$.

- *kritischer Wert* von α für e

$$\alpha_{crit}: KK_T(E_e, \alpha_{crit}) = KK_T(\{e\}, \alpha_{crit})$$

➡ für $\alpha \geq \alpha_{crit}$ lohnt sich das Prunen beim Knoten e

- *schwächster Link*: Knoten mit dem minimalen Wert für α_{crit}

5.4 Pruning von Entscheidungsbäumen

Methode

- Beginne mit dem vollständigen Baum E .
- Entferne iterativ immer den schwächsten Link aus dem aktuellen Baum.
- Falls mehrere schwächste Links existieren:
 alle miteinander im gleichen Schritt entfernen.

➡ Folge geprunter Bäume $E(\alpha_1) > E(\alpha_2) > \dots > E(\alpha_m)$
 mit $\alpha_1 < \alpha_2 < \dots < \alpha_m$

- Auswahl des besten $E(\alpha_i)$
 Schätzung des Klassifikationsfehlers auf der Grundgesamtheit
 durch l -fache Überkreuz-Validierung auf der Trainingsmenge

5.4 Pruning von Entscheidungsbäumen

Beispiel

i	E _i	beobachteter Fehler	geschätzter Fehler	tatsächlicher Fehler
1	71	0,0	0,46	0,42
2	63	0,0	0,45	0,40
3	58	0,04	0,43	0,39
4	40	0,10	0,38	0,32
5	34	0,12	0,38	0,32
6	19	0,2	0,32	0,31
7	10	0,29	0,31	0,30
8	9	0,32	0,39	0,34
9	7	0,41	0,47	0,47
10
...



E_7 besitzt den geringsten geschätzten Fehler
 und den niedrigsten tatsächlichen Fehler

5.5 Skalierung für große Datenbanken

Motivation

Konstruktion von Entscheidungsbäumen
eine der wichtigsten Methoden der Klassifikation

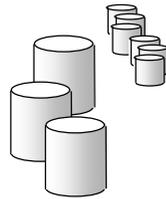
Bisher

- kleine Datenmengen
- hauptspeicherresident

Jetzt

- immer größere kommerzielle Datenbanken
- Algorithmen für sekundärspeicherresidente Daten

➡ skalieren für Datenbanken beliebiger Größe



5.5 Skalierung für große Datenbanken

Ansätze zur Skalierung

Sampling

- Stichprobe der Daten als Trainingsmenge
paßt in den Hauptspeicher
- Stichprobe aller potentiellen Splits evaluieren (bei numerischen Attributen)

🔍 Verminderung der Qualität der entstehenden Entscheidungsbäume

Unterstützung durch spezielle Daten- und Indexstrukturen

- alle Daten als Trainingsmenge
- Verwaltung auf dem Sekundärspeicher durch ein Datenbanksystem
- Einsatz spezieller Daten- und Indexstrukturen zur effizienten Unterstützung

🔍 kein Verlust an Qualität der Entscheidungsbäume

5.5 Skalierung für große Datenbanken

Unterstützung durch spezielle Daten- und Indexstrukturen

Teure Operationen

- Evaluation der potentiellen Splits und Selektion des besten Splits
bei numerischen Attributen:
 - Sortierung der Attributwerte
 - Evaluation jedes Attributwerts als potentieller Splitpunkt
- Partitionierung der Trainingsdaten
entsprechend dem gewählten Split
Lesen und Schreiben aller Trainingsdaten

➡ Growth Phase hat dominanten Anteil am Gesamtaufwand

5.5 SLIQ

Einführung [Mehta, Agrawal & Rissanen 1996]

- skalierbarer Entscheidungsbaum-Klassifikator
- binäre Splits
- Evaluierung der Splits mit Hilfe des Gini-Index
- spezielle Datenstrukturen

🔍 vermeiden das Sortieren der Trainingsdaten
für jeden Knoten des Entscheidungsbaums
und für jedes numerische Attribut

5.5 SLIQ

Datenstrukturen

- **Attributlisten**
Werte eines Attributs in aufsteigender Sortierreihenfolge
zusammen mit Referenz auf den zugehörigen Eintrag in der Klassenliste
→ sequentiell zugegriffen
→ sekundärspeicherresident
- **Klassenliste**
für jeden Trainingsdatensatz die Klasse
sowie einen Verweis auf das zugehörige Blatt des Entscheidungsbaums
→ wahlfrei zugegriffen
→ hauptspeicherresident
- **Histogramme**
für jedes Blatt des Entscheidungsbaums
Häufigkeiten der einzelnen Klassen in der Partition

5.5 SLIQ

Beispiel

Trainingsdaten

Id	Alter	Gehalt	Klasse
1	30	65	G
2	23	15	B
3	40	75	G
4	55	40	B
5	55	100	G
6	45	60	G

Klassenliste

Id	Klasse	Blatt
1	G	N1
2	B	N1
3	G	N1
4	B	N1
5	G	N1
6	G	N1

(N1)

Attributlisten

Alter	Id
23	2
30	1
40	3
45	6
55	5
55	4

Gehalt	Id
15	2
40	4
60	6
65	1
75	3
100	5

5.5 SLIQ

Algorithmus

- **Breadth-First-Strategie**
für alle Blätter des Entscheidungsbaums auf derselben Ebene
werden alle potentiellen Splits für alle Attribute evaluiert
🔍 klassische Entscheidungsbaumklassifikatoren: Depth-First-Strategie
- **Split numerischer Attribute**
für jeden Wert W der Attributliste von A (sequentieller Durchlauf)
 - bestimme den zugehörigen Eintrag E der Klassenliste;
 - sei K der Wert des Attributs "Blatt" von E ;
 - aktualisiere das Histogramm von K
basierend auf dem Wert des Attributs "Klasse" von E ;

5.5 SPRINT

Einführung [Shafer, Agrawal & Mehta 1996]

SLIQ

- Größe der Klassenliste wächst linear mit der Größe der Datenbank.
- SLIQ skaliert nur gut, wenn genügend Hauptspeicher für die ganze Klassenliste verfügbar ist.

Ziele von SPRINT

- Skalierung für *beliebig große* Datenbanken
- einfache *Parallelisierbarkeit* des Verfahrens

5.5 SPRINT

Datenstrukturen

- **Klassenliste**

keine Klassenliste mehr

zusätzliches Attribut „Klasse“ für die Attributlisten (sekundärspeicherresident)

- keine Hauptspeicher-Datenstrukturen mehr
- skalierbar für beliebig große DB

- **Attributlisten**

getrennte Attributlisten pro Knoten des Entscheidungsbaums

nicht eine Attributliste für die ganze Trainingsmenge

- keine zentralen Datenstrukturen mehr
- einfache Parallelisierung von SPRINT

5.5 SPRINT

Beispiel

Alter	Klasse	Id
17	Hoch	1
20	Hoch	5
23	Hoch	0
32	Niedrig	4
43	Hoch	2
68	Niedrig	3

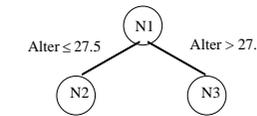
Alter	Klasse	Id
17	Hoch	1
20	Hoch	5
23	Hoch	0

Autotyp	Klasse	Id
Familie	Hoch	0
Sport	Hoch	1
Familie	Hoch	5

Autotyp	Klasse	Id
Familie	Hoch	0
Sport	Hoch	1
Sport	Hoch	2
Familie	Niedrig	3
LKW	Niedrig	4
Familie	Hoch	5

Alter	Klasse	Id
32	Niedrig	4
43	Hoch	2
68	Niedrig	3

Autotyp	Klasse	Id
Sport	Hoch	2
Familie	Niedrig	3
LKW	Niedrig	4

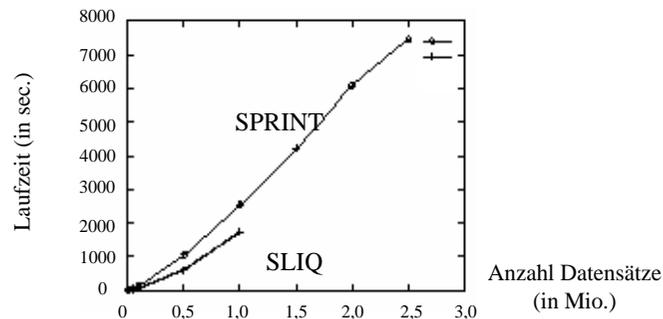


Attributlisten für Knoten N2

Attributlisten für Knoten N3

5.5 SPRINT

Experimentelle Untersuchung



- 🔍 SLIQ ist effizienter, solange die Klassenliste in den Hauptspeicher paßt für mehr als 1.000.000 Datensätze ist SLIQ nicht mehr anwendbar (Thrashing)

5.5 RainForest

Einführung [Gehrke, Ramakrishnan & Ganti 1998]

SPRINT

- nutzt den vorhandenen Hauptspeicherplatz nicht aus
- ist anwendbar nur für Entscheidungsbaum-Konstruktion mit Breitensuche

RainForest

- nutzt den verfügbaren Hauptspeicher zur Effizienzverbesserung
- für praktisch alle bekannten Algorithmen anwendbar

- 🔍 Trennung der Aspekte der Skalierung von den Aspekten der Qualität eines Entscheidungsbaum-Klassifikators

5.5 RainForest

Datenstrukturen

- *AVC-Menge* für das Attribut A und den Knoten K
enthält für jeden Wert von A ein Klassenhistogramm
für die Teilmenge aller Trainingsdaten, die zur Partition von K gehören
Einträge der Form $(a_i, c_j, zaehler)$
- *AVC-Gruppe* für das Attribut A und den Knoten K
Menge der AVC-Mengen von K für alle Attribute
- bei kategorischen Attributen: AVC-Menge wesentlich kleiner als Attributliste
→ mindestens eine AVC-Menge paßt in den Hauptspeicher
→ evtl. sogar die gesamte AVC-Gruppe

5.5 RainForest

Beispiel

Trainingsdaten

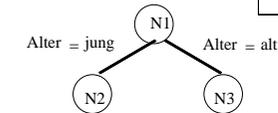
Id	Alter	Gehalt	Klasse
1	jung	65	G
2	jung	15	B
3	jung	75	G
4	alt	40	B
5	alt	100	G
6	alt	60	G

AVC-Menge Alter für N1

Wert	Klasse	Zähler
jung	B	1
jung	G	2
alt	B	1
alt	G	2

AVC-Menge Gehalt für N1

Wert	Klasse	Zähler
15	B	1
40	B	1
60	G	1
65	G	1
75	G	1
100	G	1



AVC-Menge Gehalt für N2

Wert	Klasse	Zähler
15	B	1
65	G	1
75	G	1

AVC-Menge Alter für N2

Wert	Klasse	Zähler
jung	B	1
jung	G	2

5.5 RainForest

Algorithmen

Annahme

- die gesamte AVC-Gruppe des Wurzelknotens paßt in den Hauptspeicher
- dann passen die AVC-Gruppen jedes Knotens in den Hauptspeicher

Algorithmus *RF_Write*

- Aufbau der AVC-Gruppe des Knotens K im Hauptspeicher
sequentieller Scan über die Trainingsmenge
- Bestimmung des optimalen Splits des Knotens K
mit Hilfe seiner AVC-Gruppe
- Lesen der Trainingsmenge und Verteilen (Schreiben) auf die Partitionen
→ Trainingsmenge wird zweimal gelesen und einmal geschrieben

5.5 RainForest

Algorithmen

Algorithmus *RF_Read*

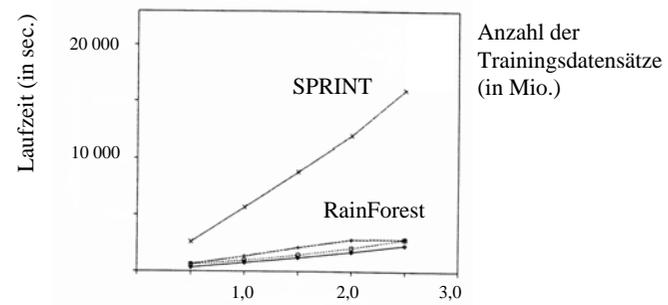
- vermeidet das explizite Schreiben der Partitionen auf den Sekundärspeicher
- Lesen der gewünschten Partition aus der gesamten Trainingsdatenbank
- gleichzeitiger Aufbau der AVC-Gruppen für möglichst viele Partitionen
→ Trainingsdatenbank wird für jede Ebene des Baums mehrfach gelesen

Algorithmus *RF_Hybrid*

- Nutzung von *RF_Read*, solange die AVC-Gruppen aller Knoten der aktuellen Ebene des Entscheidungsbaums in den Hauptspeicher passen
- Dann Materialisierung der Partitionen mit *RF_Write*

5.5 RainForest

Experimentelle Untersuchung



für alle RainForest-Algorithmen wächst die Laufzeit linear mit n
RainForest ist wesentlich effizienter als SPRINT