

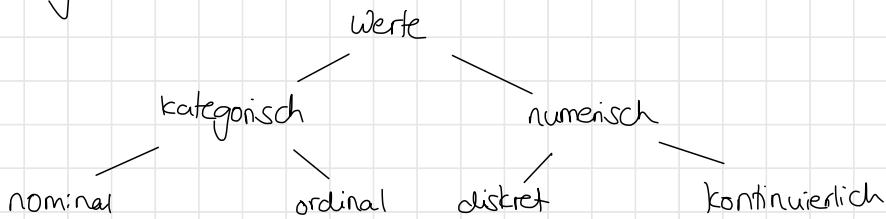
AGD 1

WS 19/20

Statistik

Datenbeschaffenheit

Kategorisierung



- nominal: keine natürliche Ordnung der Werte
z.B. Farben, Namen
- ordinal: Ordnung gegeben
z.B. gestern < heute < morgen
- diskret: endliche Anzahl möglicher Werte
- kontinuierlich: Anzahl mögl. Werte nicht abzählbar

Dimensionalität

- univariant: eindimensional (z.B. Alter)
- multivariant: multidimensional (z.B. x-y-Koord.)
- hochdimensional: Daten mit sehr vielen Dimensionen (z.B. Kunden)
↳ erfordern für viele Datenanalyseprobleme spez. Lösungen
- ohne Dimensionalität
→ oft: Abstände zw. Objekten berechenbar "metrische Daten"

Metrische Daten

in metrischen Räumen geg.: Domäne M , Metrik d mit Eigenschaften
 $\forall p, q, r \in M$:

- Symmetrie: $d(p, q) = d(q, p)$
- Definitheit: $d(p, q) = 0 \Leftrightarrow p = q$
- Dreiecksungleichung: $d(p, r) \leq d(p, q) + d(q, r)$

Einfache deskriptive Statistik

Aggregatsfunktionen

Aggregate: kombiniere alle Werte eines Attributs in einen Skalarwert
z.B. Count, sum, min, max, avg

Klassifizierung sei $F = \text{Aggregatsfunktion}$

- distributiv: $\exists \text{Funktion } G: F(\{x_{ij}\}) = G(\{F(\{x_{ij} | i=1, \dots, I\}) | j=1, \dots, J\})$
z.B. min, max, count

- algebraisch: $\exists \text{Funktion } G, \text{ die M-Tupel liefert und Funktion } H:$
 $F(\{x_{ij}\}) = H(\{G(\{x_{ij} | i=1, \dots, I\}) | j=1, \dots, J\})$

- M-Tupeltyp a priori bekannt
z.B. avg, truncated avg

- holistisch: Funktionen die weder algebraisch noch distributiv sind
→ keine Speicherbedarf beschr. für Sub-Aggregate ($\{x_{ij} | i=1, \dots, I\}$)
z.B. häufigster Wert, median

→ distributiv, algebraisch parallel ausführbar

- self-maintainable: nach Einfügen / Löschen kann neuer Wert aus altem Wert und Änderung berechnet werden
z.B. max, count

Kennzahlen und Darstellungen

- arithmet. Mittel ("Durchschnitt") $\bar{x} = \frac{1}{n} \sum_i x_i$

- gewichtetes arithmet. Mittel $\bar{x}_w = \frac{\sum_i w_i x_i}{\sum_i w_i}$

- midrange: Mitte des Wertebereichs: $\frac{\max - \min}{2}$

- Median: mittlerer Wert bei sortierten Werten
 ↗ der beiden mittleren wenn # Werte gerade
 → anwendbar auch auf ordinale Daten, aber nicht auf nominale Daten → erfordert Ordnungskriterium

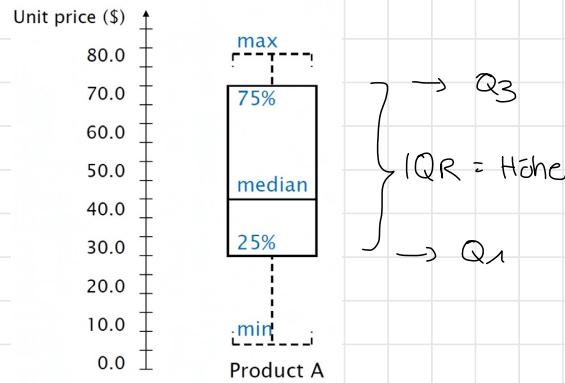
- Modalwert / Modus: häufigster Wert (nicht zwingend eindeutig)
 - schwierig für kontinuierliche Daten
 - jeder Wert ex. nur einmal: Modus undefined

- Quartile: $Q_1 = \text{oberste } 25\%$
 $Q_3 = \text{oberste } 75\%$.
 $IQR = (\text{Inter-Quartile-Range}) = Q_3 - Q_1$

- Ausreißer: zB Werte die mehr als $1,5 \cdot IQR$ von Q_1 nach unten / Q_3 nach oben abweichen

$$\min \leq Q_1 \leq \text{median} \leq Q_3 \leq \max$$

↪ Darstellung durch Boxplots



- Varianz: $\sigma^2 = \frac{1}{n-1} \sum_i (x_i - \bar{x})^2$

- Standardabweichung: $\sigma = \sqrt{\sigma^2}$

- Histogramm: x -Achse $\hat{=}$ Attributwert
 y -Achse $\hat{=}$ Häufigkeit
 bzgl. Bucket
-] Aufteilung Wertebereich in Buckets (Intervalle)

- Equi-Width Histogramm: Intervallbreite gleich
- Equi-Depth Histogramm: Summe der Häufig. aller Buckets gleich
- - ungeeignet für viele Dimensionen
keine Antwortverfeinerung

Entropie = wie zufällig sind Daten einer Menge verteilt
Maß für die Unordnung

$$H(S) = -\sum_j p_j \cdot \log p_j$$

p_j = relative Häufigkeit der Klasse j in S
 $\log p_j$ = statistische Signifikanz \rightarrow wie überraschend ist Ergebnis
 \sim falls $p=1 \sim \log(1)=0$

- $H(S)$ minimal falls $p_1 = 1$
- $H(S)$ maximal falls $p_i = p_j \quad \forall i, j \in S$

Joint Entropy Verallg. für mehr als 1 Variable

$$H(X, Y) = -\sum_x \sum_y p(x, y) \cdot \log_2 [p(x, y)]$$

Wahrscheinlichkeitstheorie

Wahrscheinlichkeitsraum (Ω, \mathcal{F}, P)

Ω = Ergebnismenge

$\mathcal{F} \subseteq 2^\Omega$ = Ereignisraum

P = Wahrscheinlichkeitsmaß \rightarrow ordnet jedem $A \in \mathcal{F}$ einen Wert aus $[0, 1]$ zu

- ~ \mathcal{F} enthält triviales Ereignis
- \mathcal{F} enthält leeres Ereignis
- \mathcal{F} abgeschlossen unter Vereinigung, Komplement

\rightsquigarrow P erfüllt Axiome:

- Nichtnegativität: Für alle $a \in \mathbb{F}$: $P(a) \geq 0$
- triviales Ereignis: $P(\emptyset) = 1$
- Additivität: $\forall a, b \in \mathbb{F} \text{ und } a \cup b = \emptyset : P(a \cup b) = P(a) + P(b)$

Zufallsvariable

• Größe deren Wert vom Zufall abhängt

Multivariate Verteilungen, Randverteilungen

Multivar. Verteil. = Wahrscheinlichkeitsverteil. einer mehrdim. Zufallsvariable

ZB		P	
		$x=1$	$x=2$
$y=0$	$1/4$	$1/4$	
	$1/4$	$1/4$	

$P(x=a, y=b)$ ist Wkt., $P(x, y)$ ist multivariate Wahrscheinl. v. v.

$P(x), P(y)$ = Randverteilungen

$$\rightsquigarrow P(x) = \sum_{b \in \text{Val}(y)} P(x, y=b)$$

Bedingte Verteilungen

Verteilung einer ZV abh. vom (bekannten) Wert einer anderen ZV

$$P(X=a | Y=b) = \frac{P(X=a, Y=b)}{P(Y=b)}$$

Unabhängigkeit

Verteilung von X unabh. von Verteilung von Y

$$\rightsquigarrow \text{also falls } P(x) = P(x|y)$$

und dann gilt:

$$P(x, y) = P(x) \cdot P(y)$$

diskrete ZV: Wktfunktion

$$f(x) = P(X=x) \quad \forall x \in \mathbb{R} \quad (\text{unabh. davon ob } X \text{ diskret})$$

$$f \geq 0$$

$$\sum_x f(x) = 1$$

kontinuierliche ZV: Wktfunktion

$$\text{Dichtefunktion: } P(X \in [a,b]) = \int_a^b f(x) dx$$

\rightsquigarrow Intervalle statt Werte denn $P(X=x)=0$

$$f \geq 0$$

$$\int_{-\infty}^{\infty} f(x) dx = 1$$

Erwartungswert

$$E(x) = \sum_{a \in \text{Val}(x)} a \cdot P(X=a)$$

$$E(x) = \int_{\text{Val}(x)} x \cdot f(x) dx$$

Varianz

$$\text{Var}(x) = E((x - E(x))^2) = E(x^2) - E^2(x)$$

Zusammenhänge zwischen Variablen

Kovarianz

! nicht normiert \rightsquigarrow Werte ohne Kontext nicht vergleichbar

x, y sind ZV

$$\rightsquigarrow \text{Cov}(x,y) = E[(x - E(x)) \cdot (y - E(y))]$$

$$\cdot \text{Cov}(x,x) = \text{Var}(x)$$

· Veranschaulichung: x groß gdw. y groß $\rightsquigarrow \text{Cov}(x,y) > 0$

x klein gdw. y groß $\rightsquigarrow \text{Cov}(x,y) < 0$

kein monotoner Zsm. hang $\rightsquigarrow \text{Cov}(x,y) = 0$

· Änderung einer Skala ändert Cov-Wert

Abgrenzung zu diversen Korrelationsmaßen: Korrelation normiert auf $[1,1]$

Kovarianzmatrix

- Zufallsvektor der Länge n gegeben \rightsquigarrow Matrix der paarweise Kovarianzen der Elemente des Zufallsvektors = Kovarianzmatrix
- auf Diagonale: Varianzen, Matrix ist symmetrisch
- möglicher Input für PCA

Statistische Tests

Chi-Quadrat-Test

hier: Unabhängigkeitstest

- Sind Z-Verteilungen unabhängig \rightarrow ZV ordinal od. diskret!

1) Hypothese aufstellen: X, Y unabhängig

2) Beobachtete Häufigkeit in Tabelle aufschreiben $\rightsquigarrow h_{n,m}$

	x_1	x_2	x_3	...	Σ_i
y_1	$h_{1,1}$	$h_{1,2}$	$h_{1,3}$.
y_2	$h_{2,1}$
:
Σ_j

3) Erwartete Häufigkeit in Tabelle aufschreiben $\rightsquigarrow \hat{h}_{n,m}$

	x_1	x_2	x_3	...	Σ_i
y_1	$\hat{h}_{1,1}$
y_2	$\hat{h}_{2,1}$
:
Σ_j

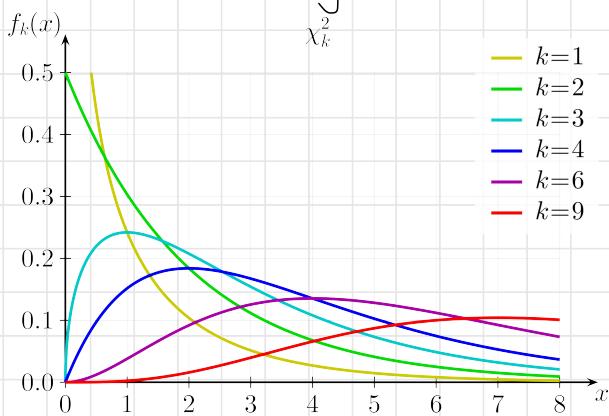
$$\hat{h}_{n,m} = \frac{h_{n,*} \cdot h_{*,m}}{N}$$

4) Prüfgröße berechnen

$$\chi^2 = \sum_n \sum_m \frac{(h_{n,m} - \hat{h}_{n,m})^2}{\hat{h}_{n,m}}$$

5) Freiheitsgrad ermitteln $k = (n-1) \cdot (m-1)$

6) Wahrscheinlichkeit, dass χ^2 diesen Wert annimmt geg. k :
aus χ^2 -Verteilung ablesen



→ Hypothese dass Verteilung unabhängig zurückweisen falls abgelesene Wkt $\leq \alpha$ (Schwellenwert z.B. $\alpha = 0,01$)

⚠ falls Hypothese nicht zurückgew.: kein Beweis, dass X_1, X_2 wirklich unabhängig! → keine verlässliche Aussage in diesem Fall

Kolmogorov - Smirnov - Test

- stimmen 2 Verteilungen überein
- keine Beschränkung, dass ordinal od. diskret

1) Nullhypothese $H_0: F_X(x) = F_0(x)$

2) Vergleich der empirischen Verteilungsfunktion F_n mit F_0
($F_n(x)$ ist Summe aller relativen Vorkommen von Werten $\leq x$)

Teststatistik: $d_n = \|F_n - F_0\| = \sup_x |F_n(x) - F_0(x)|$

Wilcoxon - Mann Whitney Test

- 2 Verteilungen mit Verteilungsfunktionen F_X, F_Y

$$F_Y(x) = F_X(x-a)$$

- Unabh. Stichproben X_1, \dots, X_m und Y_1, \dots, Y_n

$$H_0: a=0$$

$$H_1: a \neq 0$$

→ ist Abweichung der Mediane statistisch signifikant

Rangsummenstatistik: Ordne Zahlen von X, Y gemeinsam aufsteigend

→ niedrigster Wert hat Rang 1

→ Summe der Ränge von X und von Y berechnen

→ berechne Kennzahl, vergleiche mit Tabelle

! wenn H_0 zurückgewiesen wird muss H_1 nicht getestet werden

wenn keine Hypothese zurückgewiesen wird → keine Aussage möglich

Bernoulli - Experiment

N Datenobjekte

p = Erfolgswkt. einzelnes Experiment

S = # erfolgreicher Experimente

beobachtete Erfolgsquote $f = \frac{S}{N}$ ist ZV

$$\text{Varianz: } \text{Var} = \frac{p(1-p)}{N}$$

→ Frage: geg. f, wie groß ist p wahrscheinlich? zB mit 95%iger Sicherheit ist $p \in [..]$

→ überführe f in ZV mit Mittelwert 0, std 1

$$\Pr \left[-z < \frac{f-p}{\sqrt{p(1-p)/N}} < z \right] = c$$

→ bestimme p für geg. c und z

→ Ungleichung als Gleichung schreiben, Lösungen = Intervallgrenzen

Wahrscheinlichkeit einer bestimmten Folge von Ausgängen von Bern.-Exp:

$$\prod_{i=1}^n p^{y_i} (1-p)^{1-y_i} \quad y_i \in \{0,1\}, \text{Ausgang } i\text{-tes Exp.}$$

$$= \sum_{i=1}^n (1-y_i) \log(1-p) + y_i \log(p)$$

Diskussion stat. Tests

- ⊕ · Werkzeug für Anwendungsentwickler
- gute Kombinationsmöglichkeit mit anderen DB-Features

Datenreduktion

Datenmengen im TB/PB-Bereich \rightarrow komplexe Analysen aller Daten
quasi unmöglich

\rightarrow Datensimplifikation = Repr. des Datenbestandes mit deutlich geringerem Platzbedarf bei (fast) gleicher Analysemöglichkeit

Numerosity Reduction

- Parametrische Verfahren:
 - Annahme: Datenverteilung folgt Modell
 \hookrightarrow schätzt Modellparameter, speichere nur diese + ggfs. Ausreiser
- nicht-parametrische Verfahren:
 - keine Modell-Annahme

Sampling

nicht-param.

- Arbeitet mit repräsentativem Ausschnitt aus Datensetze

Feature Selection

- Anwahl Teilmenge der Attribute
- "von Hand" oder automatisch
 - Kriterium: fehlende Attributwerte anhand vorhandener Attribute möglichst gut vorhersehbar
 - z^d mögliche Teilmengen von Attributen \rightarrow Heuristiken
 - behalte Attribute mit bester Aussagekraft
 - Schrittweise Attribute hinzufügen
 - oder
 - Schrittweise eliminieren von Attr. mit geringster Aussagekraft

Principal Component Analysis (PCA)

= Hauptachsentransformation / Singularwertzerlegung

- geg.: N k-dim. Datenobjekte
- finde Hauptachsen: $c \leq k$ orthogonale Vektoren die Datenbestand am besten repräsentieren

$$A = U S V^T$$

. . .

A $m \times n$ Attrib \times Obj	U $m \times r$ Spalten- orthonormal	S $r \times r$ diagonal $r \leq \min(m, n)$	V^T $r \times n$ Zeilen- orthonormal
--	--	--	---

V = Eigenvektoren von $A^T A$ (Ähnlichkeitsmatrix bzgl. X)

$U = \dots$ Ähnlichkeitsmatrix bzgl. Y

→ weglassen der kleinsten Diagonalelemente
 Sortieren der Reihenfolge der Koord. Achsen absteigend
 Reduktion von S auf $S_k \in k \times k$

$$A_k = U_k$$

$$m \times n \quad m \times k$$

Spalten-
orthonormal

$$S_k$$

$$k \times k$$

diagonal
 $r \leq \min(m, n)$

$$V_k^T$$

$$k \times n$$

Zeilen-
orthonormal

Zeilen von V_k^T beschreiben unkorrelierte Hauptrichtungen im Raum

Clustering nicht-param.

Identifizierung von Gruppen von Items die nahe beieinander liegen
 → Ausschnitte im Raum mit überdurschn. vielen Datenelementen

• speichere nur Cluster (z.B. als cluster Features: Mittelpunkt, ...)

① Diskretisierung

- geg. Objekte aus versch. Klassen
- vergrößere Attributwerte → Intervallgrenzen berechnen?

entropiebasiert Entropie des Splits berechnen top-down

• geg. Datenbestand S mit Attribut entlang dessen diskr. werden soll

→ Partitionierung in S_1, S_2 mit Begrenzung T

$$\text{Entropie}_{\text{Split}}(S, T) = \frac{|S_1|}{S} \text{Entropie}(S_1) + \frac{|S_2|}{S} \text{Entropie}(S_2)$$

- Suche Abgrenzung die Entropie minimiert
- Rekursiv wdh. bis Abbruchkrit. erreicht

merge-basiert

bottom-up

finde 'beste' benachbarte Intervalle und merge
→ wdh. bis Abbruchkrit. erreicht

χ^2 -Analyse : Bewertung versch. merges

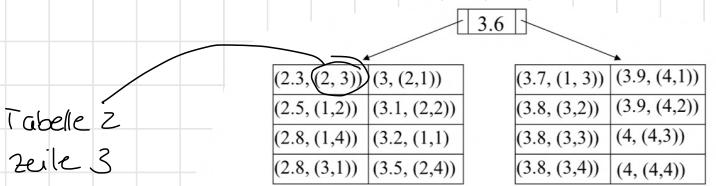
- jeder Wert zunächst eigenes (Interval)
- χ^2 -Test für alle Paare benachbarter Intervalle
- merge Paare mit kleinstem χ^2 -Wert
- wdh. bis Abbruchkrit. erreicht

Informatik Grundlagen : Räumliche Indexstrukturen

→ für Clustering, Outlier Detection, Klassifikation

Index

- Seitenweise Anordnung der Daten
↳ Einheit des Zugriffs : Seiten
- Daten müssen im Hauptspeicher vorliegen damit arbeiten
damit möglich
- Zugriffslücke : Seiten in Hauptspeicher laden zu teuer



← Indextree für Note
verhindert laden jeder
Seite um z.B. Note 1,0
zu suchen

Tom, 20, 3.2, EE	Mary, 24, 3, ECE	Lam, 22, 2.8, ME	Chris, 22, 3.9, CS
Chang, 18, 2.5, CS	James, 24, 3.1, ME	Kathy, 18, 3.8, LS	Vera, 17, 3.9, EE
Bob, 21, 3.7, CS	Chad, 28, 2.3, LS	Kane, 19, 3.8, ME	Louis, 32, 4, LS
Pat, 19, 2.8, EE	Leila, 20, 3.5, LS	Martha, 29, 3.8, CS	Shideh, 16, 4, CS

- Index für mehrere Attribute möglich
! Index für (name, note) != (note, name)

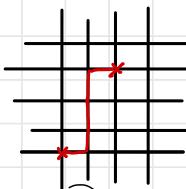
Normalisierung der Attribute

- Abstand \sim Unähnlichkeit von Datenobjekten
- üblicherweise: Dim. haben versch. Einheiten
→ Abstandsmetrik?

Normalisierung: $a_i = \frac{v_i - \min v_j}{\max v_j - \min v_j}$

Manhattan-Distanz

$$d(x_1, x_2) = \sum_{i \in D} |x_{1,i} - x_{2,i}|$$



- Summe der Abstände in einzelnen Dimensionen
- je mehr Dim. desto größer d

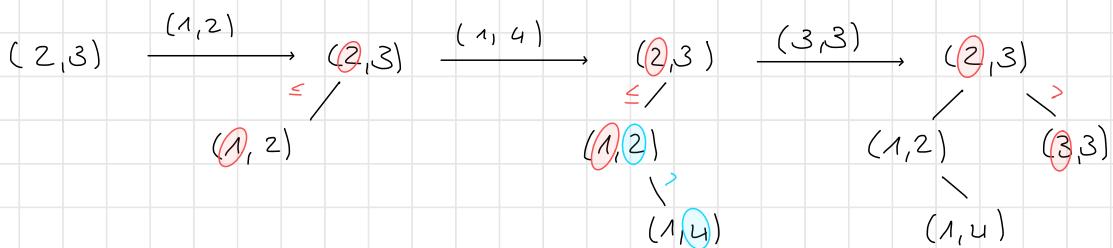
Manhattan-Segmental-Distanz

normalisiert Anzahl Dim. heraus

$$d_D(x_1, x_2) = \frac{\sum_{i \in D} |x_{1,i} - x_{2,i}|}{|D|}$$

Kd-Baum

Aufbau: Füge Punkte ein durch Split, Vergleich in Dim.
 \leadsto eine Dim. nach der anderen



Datenraum: 1. Split entlang Gerade durch $x = 2$
Split im linken Teil durch $y = 2$

NN - Berechnung \rightarrow finde NN von x

- Traversiere Baum als sollte x eingefügt werden
- Blatt b erreicht: current-best = dist(b, x)
- gehe Baum rückwärts nach oben:
 - an jedem Knoten k :
 - dist(k, x) < current-best ? update
 - checke anderen Subtree:
 - ist Abstand der split-Dim - koord. von x und von k < current-best ?
 - \rightarrow falls ja: absteigen- Fertig wenn am Wurzelknoten angekommen

Objekte mit räumlicher Ausdehnung \rightarrow Minimum Bounding Rectangle

\rightarrow Frage: In welche Rechtecke fällt Anfragepunkt

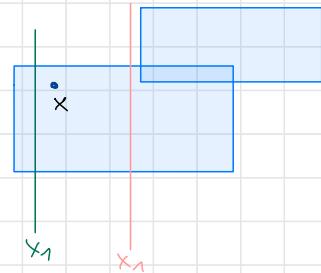
Rechteck def. durch $(\text{links}, \text{rechts}, \text{unten}, \text{oben}) = (x_1, x_2, y_1, y_2)$

Anfragepunkt (x, y)

Fall 1: $x < x_1 \rightarrow$ nur links absteigen

Fall 2: $x \geq x_1 \rightarrow$ in beide Teilbäume absteigen

$(x_1, -, -, -)$
/ \
 $(-, x_{21}, -, -)$ $(-, x_{22}, -, -)$



Kd_B-Baum

- Kd_B-Baum unbalanciert
- kombiniere mit B^{*}-Baum
- Unterschiede physische und logische Knoten
- physischer Knoten enthält mehrere logische Regionen
- (physischer) Knoten nicht mehr pro Split-Dimension

⊕ Abstand Wurzel, Daten immer gleich
pro phys. Knoten eine Speicherseite
↳ effizienter Zugriff

⊖ komplexe Reorganisation

R-Baum

Einfügen

Fall 1: Punkt fällt in Zone genau eines Kindknotens

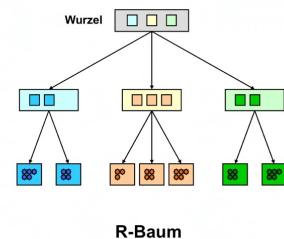
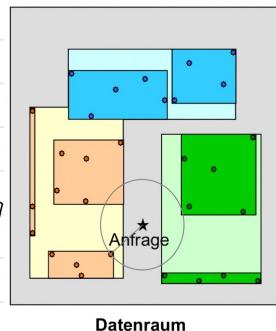
Fall 2: Punkt fällt in Überlappung zweier Kindknoten

Fall 3: Punkt fällt in Zone keines Kindknotens

Fall 3 Lösung: Einfügen in Kind dass ein Fläche sich am wenigsten vergrößert

Split falls Kindknoten zu viele Datenobjekte enthält

Finde 2 Kinder mit max. Abstand, finde Achse mit max. Abstand → Sortiert mit der Kinder



Instanzbasiertes Lernen

- bisher: Step 1: konstruiere explizites Modell
Step 2: danach Anwendung des Modells auf neue Datenobjekte
- Instanzbasiert: kein Step 1 \leadsto "Lazy Learning"
- NN-Suche zur Laufzeit
 \leadsto Suchbaum benötigt
- + gut bei veränderlichen Datenbeständen
jedes Attribut hat gleichen Einfluss auf Resultat
- Problem der hohen Dimensionalität
Noise problem. falls $k > 1$

Klassifikation mit Entscheidungsbäumen

Situation Supervised Learning:

- lerne Modell M aus Trainingsdaten
- wende M auf neue Objekt an

Trainingsdaten: $T = \{(x_1, y_1, \dots, l_1), (x_2, y_2, \dots, l_2), \dots, (x_n, y_n, \dots, l_n)\}$
 $\rightarrow l$ ist Label/Klassen zugehörigkeit

Attribut = einzelne Werte x_i, y_i, \dots

- i. A. für Klassifikation bedingt hilfreich

Feature = Funktion die Attributwerte auf einen Wert abbildet

- i. A. sehr viele Features möglich
 \hookrightarrow überlege "nützliche Features" \rightarrow Idee: Classifier sucht sich dann die besten aus

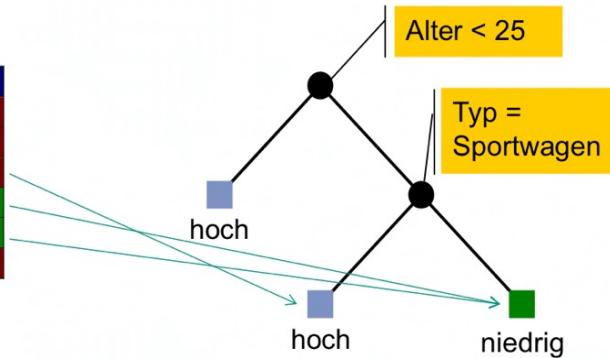
$$\Rightarrow T' = \{ (f_1(x_1, y_1), f_2(x_1, y_1), \dots, f_m(x_1, y_1), l_1), \\ (f_1(x_2, y_2), \dots, f_m(x_2, y_2), l_2), \\ \dots \}$$

Binäre Entscheidungsbäume

Baue Baum auf basierend auf Trainingsdaten
→ Split - Attribut? Schwellenwerte?

- Trainingsdatum = Blatt
- Vorhersagewert = Häufigstes Klassenlabel im Datenbestand der Blatt zugeordnet ist

Alter	Typ	Risiko
23	Familie	Hoch
17	Sportwagen	Hoch
43	Sportwagen	Hoch
68	Familie	Niedrig
32	LKW	Niedrig
20	Familie	Hoch



Split finden

Entropie eines Splits: $E(S_1, S_2) = \frac{n_1}{n} E(S_1) + \frac{n_2}{n} E(S_2)$
→ wähle Split der Entropie minimiert (E nahe bei 0)

Aufbau Entscheidungsbäume

split (Datenbestand D) {
if D homogen: break;

! noch kein Overfitting handling

Finde Split mit niedriger Entropie für D

- wähle Split - Attribut A
- wähle Split - Wert S_A

$$D_1 := \{d \in D : d_A < S_A\}$$

$$D_2 := D \setminus \{d \in D : d_A \geq S_A\}$$

split (D_1)

split (D_2)

Pruning

Overfitting = Entscheidungsbaum zu sehr auf T zugeschnitten
üblicherweise Postpruning

Prepruning

- belasse Knoten als Blatt wenn Split nicht mehr viel bringt
- dR schrittweise
 - ↳ Kombinationen von Attributen als gute Vorhersage nicht erkannt "

Postpruning

- baue Baum auf, dann zurückschneiden
- ⊕ Baum wird nur einmal aufgebaut
 - ↳ geringere Kosten der Modellerstellung

Abschätzen der Fehlerrate

Frage: inneren Knoten durch Blatt ersetzen?

→ Schätzt Fehlerrate des Knotens

Holdout-Daten: Daten die nicht fürs Training verwendet werden
oder: Heuristik

$$N = \# \text{ Daten im Knoten}$$

$$E = \# \text{ Errors (Label weicht vom häufigsten Wert ab)}$$

$$q = \text{Tatsächliche Fehlerrate (unbekannt)}$$

→ Berechne Intervall für q mit Bernoulli-Experiment

→ Schätzwert ist dann obere Intervallgrenze

Wahrscheinlichkeiten als Vorhersageergebnis

- i.A. nützlicher: Vorhersage $\in [0,1]$ statt 'entweder 1 oder 0'
- Entscheidungsbaum: Wkt $\hat{=}$ relative Häufigkeit der häufigsten Klasse pro Blatt

Kostenberücksichtigung

Fehler:

- False Positives: Ja vorhergesagt obwohl nein richtig wäre
- False Negatives/ Misses: umgekehrt

\rightsquigarrow False positives ggf. teurer als Misses

Lösungen : (2 Klassen im Folgenden)

- Lernverfahren min. Anzahl Fehler ohne Untersch. der Fehlerarten:
 - ↳ False positives zB 10mal teurer als Misses?
 - 10mal mehr No-Instanzen in T packen
 - Verfahren trainiert sensitiver

• Lernverfahren liefert Wkt zurück:

ζ minimiere erwartete Kosten der Vorhersage

2B FP kostet 10 Wkt für Nr.: 017

Miss kostet 1 \Rightarrow erwartete Kosten für No:

korrekt kostet 0

$$\underbrace{0,3 \cdot 10}_{\neq P} \rightarrow \underbrace{0,7 \cdot 0}_{\text{korrekt}}$$

=> Wahrscheinlichste Vorhersage i. A. nicht best... .

Bsp: FP kostet 100 True Pos. bringt 10

Kiss kostet 0 True Neg. bringt 0

Classifier liefert 51% Ja, 49% No

→ Annahme Ja bringt Gewinn

$$R(\lceil a \rfloor x) = 0,51 \cdot 10 + 0,49 \cdot (-100) = -43,9$$

→ Annahme Nein bringt Gewinn

$$R(\text{Nein}(x)) = 0$$

Conditional Risk

j = mögliche tatsächliche Klasse

i = verhängte Klasse

$$R(i|x) = \sum_j P(j|x) \cdot cost(i,j)$$

Null-Werte in T

Optionen:

1) Null wie weiteren möglichen Wert behandeln

→ Sinnvoll wenn Null Bedeutung hat

zB Null bei Abidatum \Rightarrow kein Abi

→ nicht sinnvoll bei zB Geburtsdatum fehlt

2) Bei Entscheidungsbäumen mit gewichteten Daten:

a) ignoriere Null bei Split-Findung

b) n_1 Daten haben Attributwert \leq Splitwert
 n_2 >

→ Objekt mit Null kommt mit Gewicht ...

... $\frac{n_1}{n_1+n_2}$ nach links

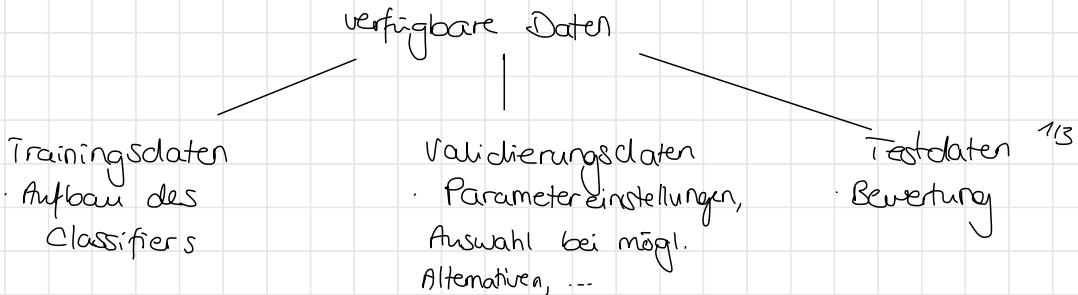
... $\frac{n_2}{n_1+n_2}$ nach rechts

→ Null-Objekte anteilig gemäß Gewicht bei vorhergesagtem Label berücksichtigen

Evaluation von Classifiern

Trainingsdaten

Resubstitution Error = Fehlerrate auf Trainingsdatenbestand



→ nach Training: nutze Testdaten zur Verfeinerung

- Stratification: Sicherstellen, dass Eigenschaften (zB Labels) etwa gleichverteilt in Partitionen
z.B. nicht trivial

Crossvalidierung

→ repeated holdout Methode:

- Zufällige Aufteilung Trainings-/Testdaten → mehrfache Wahl.
- jeweils Güte ermitteln (zB Fehlerrate), dann Durchschnitt

Crossvalidierung

- Folds = Fixe # Partitionen n, eine zum Testen, n-1 für Training
 - n Wiederholungen
 - n-fache Crossvalidierung
- Standard: n=10, stratifizierung der Klassenlabel

Fehlerarten und entsprechende Maße

Gesamtfehler besteht aus 2 Summanden: Varianz, Bias

- Bias = Fehler beim Lernen → unabh. von Größe des Datenbestands
- Varianz = Fehler durch Begrenztheit der Trainingsdaten

→ "Bias-Variance Decomposition"

Fehlerarten

- 2 Klassen:

		Vorhersage	
		yes	no
Wahrheit	yes	true positive	false negative = miss
	no	false positive	true negative

Gesamt-Erfolgsquote:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

• > 2 Klassen: Konfusionsmatrix

		Vorhersage			total
		a	b	c	
Wahrheit	a	88	10	2	100
	b	14	40	6	60
	c	18	10	12	40
total		120	60	20	200
Gesamt - Erfolgsquote =		$\frac{88+40+12}{200}$			$= \frac{140}{200} = 70\%$

Kappa-Koeffizient

wie gut ist die Vorhersage wirklich → vergleiche mit Referenz-Classif. der basierend auf Anzahl Klassenzug. schätzt, ansonsten rat

		Vorhersage			total
		a	b	c	
Wahrheit	a	60	30	10	100
	b	36	18	6	60
	c	24	12	4	40
total		120	60	20	200

→ Diagonaleinträge müssen also aus 70% rausgerechnet werden

$$K = \frac{140 - (60 + 18 + 4)}{200 - (60 + 18 + 4)} = \frac{58}{118} = 49,2\%$$

Verhältnis Verbesserung zu max. möglicher Verbesserung
↖ Nenner

Wertebereich:

- jetzt alles richtig, vorher etwas: $k=1$
- jetzt alles falsch, vorher etwas: $k \ll 0$
- vorher alles richtig: undefiniert

Problem bei direktem Vergleich: 97% vs 82%

→ absoluter Unterschied?

relativer Unterschied?

Wahrscheinlichkeiten als Ergebnis

Loss Function wie viel verliert man durch unkorrekte Vorhersage

z.B. 0-1-Loss function:
0 falls korrekte Vorhersage
1 sonst

quadratische Loss function

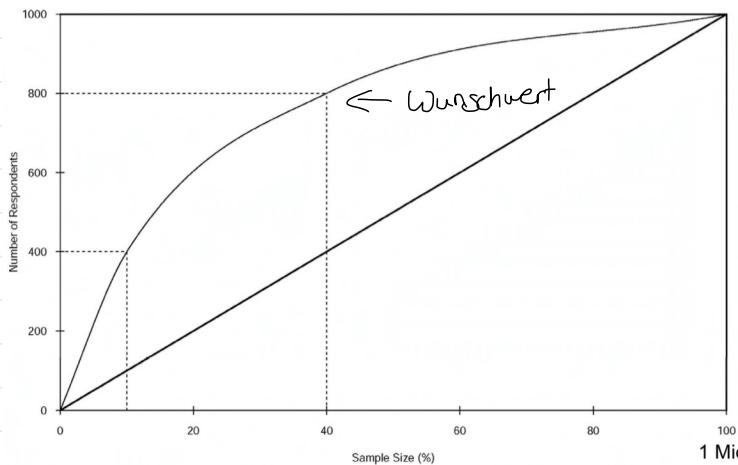
- k mögliche Vorhersagen
 - Verfahren liefert Vektor (p_1, \dots, p_k) mit Summe 1
 - tatsächliche Klassenzugehörigkeit (a_1, \dots, a_k) wobei ein $a_j = 1$ und Rest 0
- quadratic loss function: $\sum_j (p_j - a_j)^2$ [≤ 2 immer]

Informational loss function geg. (p_1, \dots, p_k) Vorhersage

- $-\log_2 p_i$:= tatsächliche Klassenzugehörigkeit
- Verfahren sollte nie Schätzung $p_i = 0$ abgeben

Lift Charts

- Verbesserung vom Modell gegenüber Zufallsvorhersage
- plot mit Zufallsvorhersage und Modellvorhersage
- versch. Classifiers führen zu versch. Kurven



- Kostenberücksichtigung: Cost / Benefit Curve
gleiche x - Achse
y - Achse : Gewinn / Verlust

ROC Kurven

ROC = Receiver operating characteristics

- FP - Rate = $100 \cdot \frac{FP}{FP+TN}$ → wie oft + Yes für NO Objekte?
1 bedeutet max. schlecht
- TP - Rate = $100 \cdot \frac{TP}{TP+FN}$ → wie oft Yes für Yes Objekte?
= Recall
1 bedeutet perfect
- Precision = $100 \cdot \frac{TP}{TP+FP}$

Aufbau des Diagramms:

- Datenbestand ist absteigend nach vorhergesagter Wkt sortiert
- für k Datenobjekte ist bekannt: # falscher & richtiger Vorhersagen
z.B. für $k=5$ ist 1:4 Punkt im Diagramm
- x-Achse ist # false pos., y-Achse ist # true pos.
ideal: links oben (Kurve macht Bauch)

f-measure :

$$\frac{2 \cdot \text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

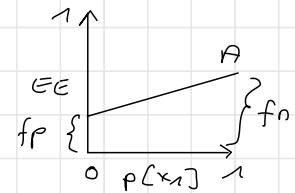
Erfolgsquote :

$$\frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

Kostenberücksichtigung

Plot: x-Achse = Probability $p[x_i] \in [0, 1]$
y-Achse = Expected Error $\in [0, 1]$

A ist bestimmter Classified: Gerade
 $fp(1-p) + p \cdot fn$



→ Kosten: A bleibt Gerade \rightarrow Summanden werden nur mit konst. Faktoren (Kosten) multipliziert

Numerische Vorhersagen: Qualitätsmaße

p_i = Vorhersage für i -tes Objekt

a_i = tatsächlicher Wert

Mean squared error

$$\frac{1}{n} \sum_{i=1}^n (p_i - a_i)^2$$

Root mean squared error

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (p_i - a_i)^2}$$

Mean absolute error

$$\frac{1}{n} \sum_{i=1}^n |p_i - a_i|$$

Relative squared error

\bar{a} = Mittelwert Trainingsdatenbestand

$$\frac{\sum_{i=1}^n (p_i - a_i)^2}{\sum_{i=1}^n (a_i - \bar{a})^2}$$

Root relative squared error

$$\sqrt{\frac{\sum_{i=1}^n (p_i - a_i)^2}{\sum_{i=1}^n (a_i - \bar{a})^2}}$$

Relative absolute error

$$\frac{\sum_{i=1}^n |p_i - a_i|}{\sum_{i=1}^n |a_i - \bar{a}|}$$

Correlation coefficient

\bar{a} = Mittelwert Testdatenbestand

$$\frac{S_{PA}}{\sqrt{S_P S_A}}$$

$$S_{PA} = \frac{\sum_{i=1}^n (p_i - \bar{p})(a_i - \bar{a})}{n-1}$$

$$S_P = \frac{\sum_{i=1}^n (p_i - \bar{p})^2}{n-1}$$

$$S_A = \frac{\sum_{i=1}^n (a_i - \bar{a})^2}{n-1}$$

* Qualitätsmaß, kein Fehlermaß
groß wenn p_i, a ähnlich

Minimum Description length

Beschreibung für Regelmäßigkeiten in Daten
↪ so kurz wie möglich

Code

$X = \text{Alphabet}$

Code C = injektive Abbildung (eindeutige Abb.) von X auf $\{0,1\}^*$
 $L_C(x)$ = Länge in Bits der Beschreibung von x

optimaler Code? geg. Wktverteilung über X

es gibt Code C_p über X mit
 $\forall x : L_{C_p}(x) = \lceil -\log p(x) \rceil$

→ Kleine Wkt $\hat{=}$ großen Codelängen

→ Code der Wktvert. über X entspricht ist optimal, also am kürzesten

Formal

E = Trainingsdatenbest.

T = Theorie $L[T] = \# \text{ Bits um } T \text{ darzustellen}$

$L[E|T]$ = Aufwand (# Bits) um E darzustellen geg. T
= Summe der informational loss function

→ minimiere $L[T] + L[E|T]$

$$\Pr[T|E] = \frac{\Pr[E|T] \cdot \Pr[T]}{\Pr[E]} = \text{Wkt dass } T \text{ gilt geg. } E$$

→ finde T so dass $\Pr[T|E]$ maximal

$$-\log \Pr[T|E] = -\log \Pr[E|T] - \log \Pr[T] + \underbrace{\log \Pr[E]}$$

nur abh. von E
→ egal bei vergl. von T s

Minimum Message length

geg. Datenbestand $x^n = x_1, \dots, x_n$ und Modelle M_1, M_2, M_3
→ welches M beschreibt x^n am besten? Kriterien:

- Fehler
- Modellkomplexität

→ auch Fehler codieren → bestes Modell ist Modell mit kleinstter Gesamtlänge

geg. Wktverteilung der Fehler: codiere Fehler analog zu MDL
→ häufige \Rightarrow kurzem Code

• Statistische Signifikanz ist Länge der einzelnen Codes

Codierungsaufwand Theorem von Shannon

x = Aussage → # Bits um x in M zu codieren?
 M = Modell

→ $-(log P(x|M))$ Bits

Association Rules

wie stehen Phänomene zueinander in Beziehung?

Item = einzelnes Element

Itemset = Menge von Items

Transaktion = Menge von Items die im Datenbestand tatsächlich vorkommt

ZB Item = Waren, Itemsets = versch. Kombinationen von Waren,
Transaktion = einzelner Einkauf

Support eines Itemsets I = # Transaktionen die I enthalten

Minimum Support σ = Schwellenwert für Support

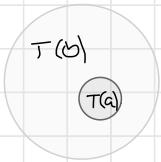
Frequent Itemsets = Itemsets mit $\text{Support} \geq \sigma$
= Mengen von Items mit pos. Korrelation wenn σ groß

Freq. Itemset maximal \Leftrightarrow nicht Teilmenge eines anderen Freq. Items

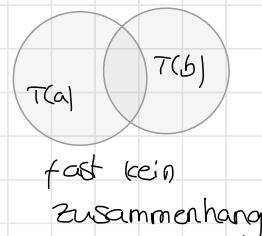
Closedness

Itemset ist closed, falls es keine echte Obermenge mit gleichem Support gibt

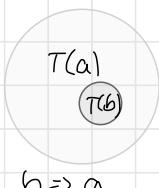
geg. Items a,b $T(a) = \{ \text{ Kunden die } a \text{ kaufen} \}$



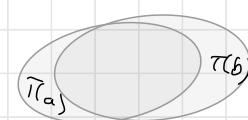
$$a \Rightarrow b$$



fast kein Zusammenhang



$$b \Rightarrow a$$



fast $a \Rightarrow b$
und $b \Rightarrow a$

$A \Rightarrow B$ support, Confidence

A, B = Itemsets

Support s von $A \Rightarrow B$ = support ($A \cup B$)
= # Transakt. die beide enthalten
 $\rightarrow T(A \cup B) = T(A) \cap T(B)$

Confidence c von $A \Rightarrow B$ = support ($A \cup B$) / support (A)
= Kunden die A, B kaufen / Kunden die A kaufen

Auswahlkriterien Association Rules

Min. Support σ
Min. Confidence γ

{ finde alle Rules mit $s \geq \sigma$ und $c \geq \gamma$

Minimum Support

hoch \rightarrow wenige Freq. Itemsets

wenige Regeln die oft vorkommen

niedrig \rightarrow viele gültige Regeln die selten vorkommen

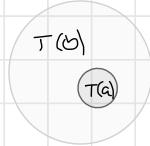
Typisch: $\sigma = 2\text{-}10\%$.

Minimum Confidence

hoch \rightarrow wenige Regeln, aber alle "logisch fast wahr"

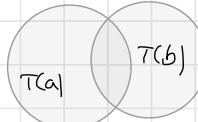
niedrig \rightarrow viele Regeln, viele sehr unsicher

Typisch: $\gamma = 70\text{-}90\%$.



$$c(a \Rightarrow b) = 100\%$$

$$c(b \Rightarrow a) \approx 0\%$$



$$c(a \Rightarrow b) \approx 0\%$$

$$c(b \Rightarrow a) \approx 0\%$$



$$c(a \Rightarrow b) \approx 100\%$$

$$c(b \Rightarrow a) \approx 100\%$$

Apriori - Algorithmus

zum Finden von Freq. Itemsets bzw. Association Rules

1) Erzeuge alle einelementigen Frequent Itemsets

2) K-mal:

a) Join

b) Prune

c) Support Counting

Ergebnis: $\cup_k L_k$ (alle L_k)

break falls $L_k = \emptyset$

(keine neuen freq. Itm.)

3) Frequent Itemsets \rightarrow Assoc. Rules

1) Einfaches Abzählen (welche Items haben support $\geq \sigma$)

2) $K = 2, \dots, n$:

a) Join: Ermittle Kandidaten C_K :

• Sortiere L_{K-1} (Freq. It. S. der Größe $K-1$)

• Finde je 2 Itemsets die sich an nur einer Stelle unterscheiden

\rightarrow zusammenfassen

b) Prune: Lösche alle Kandidaten aus C_K falls nicht alle $K-1$ elem. Teilmengen in L_{K-1} vorkommen

c) Support Counting: zähle wie oft Kandidaten vorkommen

\rightarrow zu L_K hinzufügen falls $\geq \sigma$

\rightsquigarrow Prune: prüfe nur für Kandidaten die nicht eh schon für join verwendet

Es müssen nicht alle Mengen aus L_{K-1} verwendet werden

\rightsquigarrow Prüfe ob alle Mengen abgedeckt durch Kandidaten C_K

Beispiel für Schnitt 2) a,b

Schnitt $k = 4$

$L_{k-1} = L_3$:

$\{1, 2, 3\}$

$\{1, 2, 4\}$

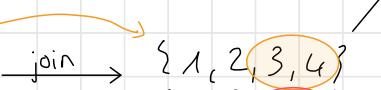
$\{1, 3, 4\}$

$\{1, 3, 5\}$

$\{2, 3, 4\}$

↪ nicht alle

Teilmengen müssen verwendet werden



$\{1, 2, 3, 4\}$

$\{1, 2, 3, 5\}$

prune

$L_4 = \{ \{1, 2, 3, 4\} \}$

Teilmengen: $\{1, 2, 3\}$ ✓ join
 $\{1, 2, 4\}$ ✓ join
 $\{1, 3, 4\}$ ✓
 $\{2, 3, 4\}$ ✓

$\{1, 3, 4\}$

$\{1, 3, 5\}$

$\{1, 4, 5\}$ ✗

$\{3, 4, 5\}$ ✗

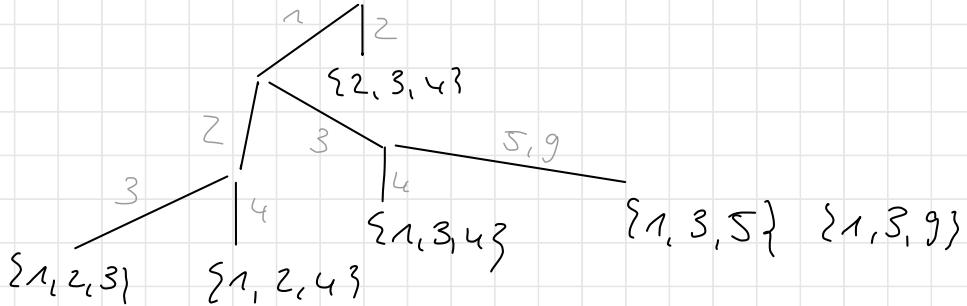
Teilmengen:
 $\{1, 3, 4\}$ ✓ join
 $\{1, 3, 5\}$ ✓ join

Support Counting: Hashtrees

Überprüfe ob Kandidat-Itemset I in Transaktion t enthalten
 → füge Kandidaten in Hashtree ein

zB Hashfkt. $\text{mod } 4$

$\{1, 2, 3\}$ $\{1, 2, 4\}$ $\{1, 3, 4\}$ $\{1, 3, 5\}$ $\{1, 3, 9\}$ $\{2, 3, 4\}$



$t = \{1, 2, 3, 5\}$

→ vergleiche jede Transaktion mit Hash-Tree

Schritt 3 : F. I. \rightarrow Assoc. Rules

$1 - \alpha = 1$ ohne α

- Betrachte alle Subsets a eines Freq. Itemsets I
- Ist $a \Rightarrow (1-\alpha)$ Association rule?
- \rightarrow hat confidence $c = \frac{s(I)}{s(a)}$
- $\rightarrow a \Rightarrow (1-\alpha)$ ist Association rule falls $c \geq \gamma$ und $s \geq \sigma$ *
- \rightarrow jede Regel wird nur einmal erzeugt
- * ist erfüllt durch Apriori-Algorithmus

⚠️ Berechnen von Association rules aufwändiger als Freq. Itemsets
 → Freq. Itemset mit Länge k hat 2^k subsets

Beispiel

$I = \{2, 3, 4\}$ [40% Support]

Subsets:

minconf=75%	{2,3}	{2,4}	{3,4}	{2}	{3}	{4}
	50%	70%	60%	80%	60%	70%

$\{2,3\} \Rightarrow \{4\}$	Support(I) = 40%	Support(a) = 50%	Confidence = 80 %	OK!
$\{2\} \Rightarrow \{3,4\}$	Support(I) = 40%	Support(a) = 80%	Confidence = 50 %	NO!

(Da I den geforderten Support hat,
 haben alle Teilmengen von I ebenfalls den geforderten Support.)

Zu aufwändig wenn nur an Freq. Itemsets interessiert:

- 1) maximale Freq. Itemsets reichen ABER: im Algo wird jede Teilmenge explizit betrachtet $\rightarrow k=30 \rightarrow 2^{30}$ Teilmengen explizit betrachtet
- 2) In jedem Schritt wird die Datenbank gescannt

Multidimensionale Association Rules

Kombiniere Werte von versch. Attributen

Multidim.

$\langle 1, \text{italian}, 50, \text{low} \rangle$
 $\langle 2, \text{french}, 45, \text{high} \rangle$

Anwendung
im
Apriori

eindim.

$\langle 1, \{\text{nat/it}, \text{age}/50, \text{inc}/\text{low}\} \rangle$
 $\langle 2, \{\text{nat/fr}, \text{age}/45, \text{inc}/\text{high}\} \rangle$

Multi-level Association Rules

Menge von Regeln auf untersch. Hierarchieebenen
z.B.



Apriori auf versch. Ebenen ausführen

- ! je weiter unten in Hierarchie desto geringer < wählen
($\geq 5\%$ aller Kunden kaufen Milch? ok)
- $\geq 5\%$ aller Kunden kaufen demeter Bio Milch? zu hoch!)

Level-Crossing Association Rules

verknüpfte Itemsets untersch. Ebenen
z.B. weißbrot => Milch

- ⚠
- hoher Support i.A. nur für abstrakte Konzepte erreichbar
 - Abstrakte Regeln aber i.A. weniger interessant
 - Problem: Menge der Freq. Itemsets wächst exponentiell mit Hierarchietiefe
≈ viele ähnliche Zusammenhänge

Berechnung Multi-Level

- Apriori pro Ebene, Beschleunigung durch integrierte Berechnung

Kodierte TA-Tabelle

Codes für versch. Items:

Milch	1	Vollmilch	11
		1,5 Milch	12
Brot	2	Vollkornbrot	21
		Baguette	22

→ T ID	Items
T ₁	{111, 121, 213, ...}
T ₂	:
T ₃	:

Itemsettabelle: Größe der Itemsets, Tiefe in Hierarchie

Frequent Itemsets schnell bestimmen

Apriori teuer:

- 1) großes k → viele Schritte über DB
- 2) zu viele Teilmengen explizit erzeugt
besonders für k = 2:
Größe C_k abh. von L_{k-1}

Weiterentwicklungen Apriori:

Hash-Filter löst Problem 2) nicht Problem 1)

$$n = \# \text{ Itemsets in } L_{k-1}$$

$$\rightarrow \text{Größe } C_k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

h Hashfunktion \rightarrow Berechne $h(x)$ für alle k -elem. Itemsets x in DB \rightarrow in Tabelle eintragen

z.B. $n = \text{mod } 7 \rightsquigarrow$ Tabelle 0|1|2|3|4|5|6

naiver Apriori: Vorkommen der Werte zählen

\rightarrow alle Werte mit $\text{sup} \geq \text{minsup}$ kommen in L_{k-1}

\rightarrow Kombinationen für C_k bilden

\rightarrow JETZT: C_k ausdünnen: betrachte alle Kand. $c \in C_k$ berechne $h(c)$

\rightarrow - falls $\text{sup}(h(c)) \geq \text{minsup}$

(also Wert $h(c)$ kommt mind. minsup mal in Hashtabelle vor)

behalte als Kandidaten

- sonst streichen

\rightarrow für jeden Schritt k neu aufbauen

$\rightarrow k$ Hashfunktionen h_1, \dots, h_k

wie groß Hashtabelle? so groß wie möglich

\rightarrow nutze Hauptspeicher möglichst gut aus

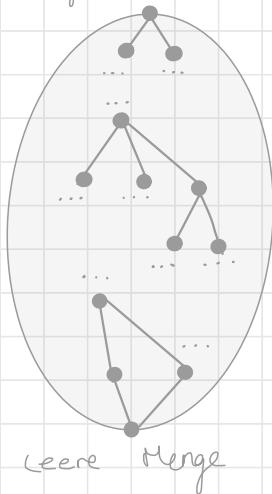
zu klein: viele Kollisionen, d.h. wenige Kandidaten werden gestrichen

Sampling

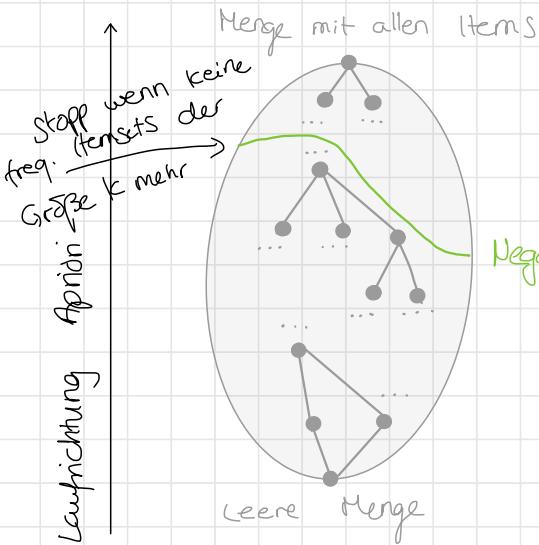
Ziel: weniger Schritte über DB (Problem 1 lösen)

Darstellung: $E_i \hat{=} \text{alle möglichen Kombinationen}$

Menge mit allen Items



Menge
an
Items
pro
Set



→ Idee: ermittle negative border für Stichprobe die in Hauptspeicher passt

- 1) neg. bord. für Support etwas größer ("zu weit unten")
- 2) neg. bord. für Support etwas kleiner ("zu weit oben")
- 3) prüfe Itemsets im Bereich der neg. border
→ nur ein DB Scan

Mathematik: Bernoulli - Experiment:

- El. Ereignis: Itemset ist in Transaktion enthalten
- berechne Konfidenzintervall für Support in Gesamtdatenbestand
- p: Wkt dass Itemset in Transaktion

Apriori-R

Problem: immer noch erzeugen der expliziten Teilmengen

Idee: in größeren Schritten nach oben laufen

↪ wenn zu weit oben wieder mit einzelnen Schritten zurück laufen

z.B. $\{1 2 3\}$ $\{1 2 4\}$ $\{1 2 5\}$ $\{1 2 6\}$ Schritt k-1
wird direkt zu

$$x = \{1 2 3 \cup 5 6\}$$

Schritt k

- falls alle Teilmengen von x erforderlichen Support haben \rightarrow top
- falls nicht: Grenze muss zwischen k-1 und k liegen
↪ also ausgehend von k-1 nach oben laufen in Schritt k
nach unten

FP - Trees

FP = frequent patterns

Apriori: Generate & Test - Paradigma

JETZT NEU: FP - Trees

Idee: baue kompakte Zusammenfassung des Datenbestands

- 1) Zähle # Vorkommen einzelner Items 1 1/2 Scans
→ behalte nur frequent Items
- Sortiere freq. Items innerhalb einer Transaktion gemäß Gesamthäufigkeit
- 2) Aufbau FP - Tree 1/2 Scan
- 3) Extrahiere Frequent Itemsets (im Hauptspeicher)

Schritt 1

Transaktion 10

100
200
300
⋮

Items
f, a, c, d, g, i, m, p
⋮

Sortierte freq. Items
c, f, a, m, p
⋮

Sortierung festlegen für Items mit gleichem Support
→ häufigste Items vorne

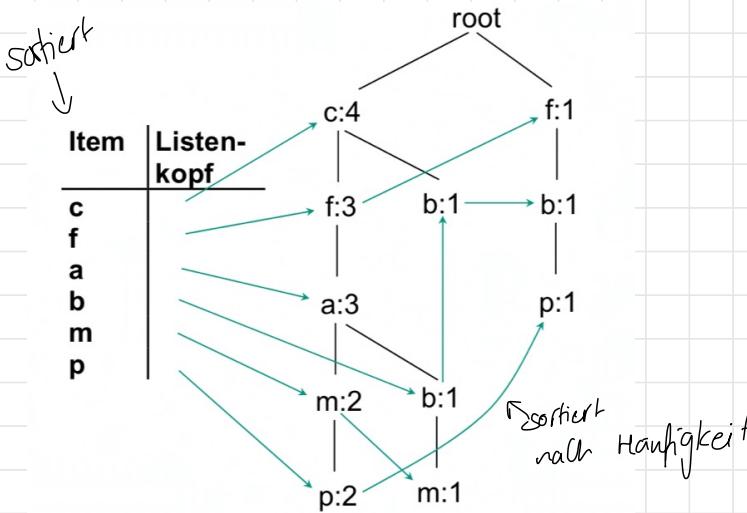
Schritt 2

Baum:

- jeder Pfad \models einer Transaktion
- Knoten = Item-ID : Häufigkeit entlang des Pfades

Header-Tabelle

- von jedem Item Zeiger auf verkettete Liste
 \hookrightarrow sortiert nach Häufigkeit



Schritt 3

- gehe Items durch: beginne mit am wenigsten häufigen Item i
- Finde Pfade die Item i enthalten
 - welche Itemkombis kommen $\geq \text{minsup}$ mal vor?
 - lösche Item i aus Baum
 - Präfix-Pfade: Pfade die mit i enden

Projected Database

DB zu groß \rightarrow FP Tree passt nicht in Hauptspeicher

\leadsto Partitioniere Transaktionen

Ausgangspunkt: sortierte Liste der freq. Items $L = \langle a:4, b:3, f:2, \dots \rangle$

- Transaktion die nur Items bis einschl. Item i enthält kommt in i -projected Database

\rightarrow Ermittle alle max. Freq. Itemsets die i enthalten (aus i -projected database)

\leadsto FP Tree für proj. DB

Dann: alle Vorkommen von i löschen, j-proj. DB in i-Tree einfügen

$\rightarrow \dots \rightarrow$ wdh. bis alle Partitionen verarbeitet

Falls i-proj. DB immer noch zu groß: Rekursion \Rightarrow feinere Partitionen.

Constrained Association Rules

Finden häufiger Teilfolgen

→ REIHENFOLGE beachten

$\langle 1, 3 \rangle$ ist TF von $\langle 1, 2, 3, 4 \rangle$ $\langle 3, 1 \rangle$ aber nicht

→ Support / conf. Berechnung analog unter Beachtung der Reihenfolge

· Im Prinzip wie Apriori:

Kandidatenerzeugung ist Self-Join von F_{k-1}

zB $\langle 1, 2, 3, 4 \rangle, \langle 2, 3, 4, 5 \rangle \rightarrow \langle 1, 2, 3, 4, 5 \rangle$

Liste von Mengen von Items

$\{a, b\}, \{b, c, d\}, \{a, f\}, \{g\}$

Teilfolge hier: versch. Definierbar

zB: TF durch Löschen von Items aus Ausgangsfolge konstruierbar

Association Rules & Constraints

· Beschränke zu findende ARs, zB ARs mit Items mit Wert $\geq 100 €$

→ nützlich: Multidim. AR

zB nur junge Leute aus DE

↪ AusgangsDB reduziert

Pruning

· Support-basiert: Kandidat eliminieren falls nicht frequent (bzw. eine Teilmenge nicht frequent)

· constraint-basiert: Kandidat eliminieren falls er ein aus den vorgegebenen constraints abgeleitetes constraint nicht erfüllt

Arten von Constraints

Data Constraints

- Einschränkung auf konkrete Werte (auch Intervalle)
zB alle Patienten die im Dez Husten hatten
(.... für 3-5 Tage)
- Einschränkung auf bestimmte Attribute des Raums

Rule Constraints

- Spezifikation der Struktur
- Spezifikation von Eigenschaften der Regeln
zB nur Freq. Itemsets der Größe 3

Query Sprache zur Formulierung von Constraints

- SQL Statement als Constraint
~ also nach Mining: sortiere Freq. Itemsets die auch Ergebnis des SQL Queries sind

⊕ effizientes Auswerten

⊖ Mining-Ergebnisse müssen genau der Struktur des Statements konform sein (Anzahl Attribute, welche Attribute)
↪ unflexibel
↪ Anfrage ala select age, X ... mit X undefiniert im Voraus nicht möglich

Meta-Rule Guided Mining

Ausgangspunkt: relationales DB-Schema

- student (name, sno, status, major, gpa, birth-date, birth-place)
- course (cno, title, dept)
- grading (sno, cno, title, instructor, semester, grade)

Data Mining Constraint: Finde alle Regeln der Form

major(s: Student, x) \wedge $Q(s, y)$ \Rightarrow $R(s, z)$ \leftarrow meta-rule
from student
where birth-place = "Canada"
in relevance to major, gpa, status

x = Platzhalter für Wert des Attributs 'major'

y =

-"-

'Q'

Q, R = beliebiges Attribut aus

Analog für R, z

Großbuchstabe = Variable für Prädikate (Attribute)

Kleinbuchstabe = Variable für entspr. Attributwert

Beispiel für gefundene Regel:

major(s, "Science") \wedge gpa(s, "Excellent") \Rightarrow status(s, "Graduate")

x

Q

y

R

z

Erlaubt: Data Constraints und Rule Constraints
birth-place = "US" zwei Prädikat links

1-var, 2-var Constraints

LHS = Left-Hand-Side

$a, b \Rightarrow c$

RHS = Right-Hand Side

zB $\sum(\text{LHS}) < 100 \wedge \min(\text{LHS}) > 20 \wedge \sum(\text{RHS}) \geq 1000$

1-var constraint

Schränkt nur eine Seite der Regel ein (LHS oder RHS)

zB $\sum(\text{LHS}) > 100$

2-var constraint

Constraint bezieht sich auf beide Seiten

zB $\sum(\text{LHS}) \leq 10 \cdot \max(\text{RHS})$

Verallgemeinerung

Bisher: Regel als Tupel mit 2 Komponenten (LHS, RHS)

Verallg.: Tupel mit n Komponenten s_1, \dots, s_n (s_i = Itemset)
zB Folge von Einkäufen

1-var Constraints Auflistung

Domain Constraints

$$\cdot S \Theta v \quad \Theta \in \{=, \neq, <, \leq, >, \geq\}$$

jedes $s \in S$ muss Constraint erfüllen
zB $\underset{S}{\text{Price}} \leq 100$

$$\cdot v \Theta S \quad \Theta \in \{\in, \notin\}$$

zB $\underset{S}{\text{snack}} \in \text{Type}$

$$\cdot S \Theta V, \forall \Theta S \quad \Theta \in \{\subseteq, \neq, \subseteq, \notin, \subset\}$$

zB $\{\text{Food}, \text{Drink}\} \subseteq \text{Type}$

Class Constraints

$$\cdot S \subseteq A \quad S = \text{Mengenvariable}, A = \text{Attribut}$$

S ist Menge aus Definitionsbereich von A

Aggregate Constraints

$$\cdot \text{agg}(S) \Theta v \quad \text{agg} = \min / \max / \sum / \text{count} / \text{avg}$$
$$\Theta \in \{=, \neq, <, \leq, >, \geq\}$$

Mining von Association Rules unter Constraints

Postprocessing

Naive Lösung: mit Apriori alle freq. Itemsets bestimmen,
dann für alle prüfen ob Constraints erfüllt

→ Optimierung: versuche Constraints möglichst früh auszuwerten
↪ Constraints "möglichst tief in Apriori reindrücken"

Anti-Monotonizität + Succinctness

↪ funktionieren gut mit Apriori

Anti-Monotonizität Ziel: Constraints möglichst früh überprüfen

1-Var Constraint ist anti-monoton \Leftrightarrow

↪ Mengen S, S' : $S' \subseteq S$ und S erfüllt $C \Rightarrow S'$ erfüllt C

D.h. jede Teilmenge erfüllt das Constraint

→ interessant ist Gegenrichtung: S' erfüllt C nicht $\Rightarrow S$ erfüllt C nicht
↪ Obermengen von S' müssen mit Apriori nicht betrachtet werden

anti-monoton

$S \ominus v \quad \ominus \in \{=, \leq, \geq\}$

$S \subseteq V$

$\min(S') \geq v$

$\max(S') \leq v$

$\text{Count}(S') \leq v$

$\text{Sum}(S') \leq v$

alle freq. constraints

nicht anti-monoton

$v \in S$

$V \subseteq S$

$\min(S') \leq v$

$\max(S') \geq v$

$\text{Count}(S') \geq v$

$\text{Sum}(S') \geq v$

$\text{avg}(S) \ominus v \quad \ominus \in \{=, \leq, \geq\}$

teilweise a: m

$S' = V$

$\min(S') = v$

$\max(S') = v$

$\text{Count}(S') = v$

$\text{Sum}(S') = v$

teilweise: kommt drauf an wie die Menge das Constraint verletzt
 $\max(S') = v, S' \subset S \rightarrow \max(S') > v \rightarrow S$ nicht mehr betrachtet
 $\max(S') < v \rightarrow$ das gilt nicht

Succinctness

Constraint ist succinct, wenn man alle Hemsets die es erfüllen explizit aufschreiben / erzeugen kann
→ statt Apriori zur Erzeugung: "von Hand" erzeugen und ein Mal Support Counting

Allgemeine Constraints

Jetzt: Constr. ist nicht anti-monoton oder succinct
→ wie in den Apriori einbauch?

Algorithmus: Schritt k generiert Menge der k -elem. freq. Hemsets
 $L_k \rightarrow$ jetzt: die constraint R erfüllen

Support-basiertes Pruning

geg. Kandidat c der Länge k
→ prüne c wenn $k-1$ elem. Teilfolge von c die R erfüllt nicht frequent ist
(also nicht in L_{k-1} enthalten)

⊖ strenge Constraints (R sehr selektiv) die nicht anti-monoton
↪ viele Teilmengen erfüllen Constraints nicht → kein gutes Pruning

→ Verbesserung durch schwächere Constraints zB $(ab)^*$ $\sim (alb)^*$
bringt nur was wenn dadurch Constraint anti-monoton / succinct

Constraint-basiertes Pruning

prüne Kandidat c falls $c \in R$ nicht erfüllt

sehr selektive Constraints: constr. basiert \gg support-basiert
denn: sup.-bas. betrachtet alle TM der Größe $k-1$ in L_{k-1}
 \sim evtl. nur wenige vorhanden

Alternativen falls R nicht anti-monoton

- naives Postprocessing
- Succinctness
- kombiniere sup.-bas., constr.-bas. Pruning mit ursprüngl. Constraint
- Pruning mit schwächerem Constraint
↳ Unterscheidung: ist schwächeres Constr. anti-monoton?

Clustering

Geg.: Datenmenge mit N d-dim. Data Items

Finde:

- Natürliche Partitionierung der Daten in k Cluster und noise
- Cluster:
 - maximiere Intra-Cluster Similarity:
Items im Cluster möglichst gleich
 - minimiere Inter-Cluster Similarity:
Items versch. Cluster möglichst verschieden

=> Ziel: optimiere Criterion Function (Bewertet Clustering)

Criterion Function für Metric Spaces:

$$E = \sum_{i=1}^k \sum_{x \in C_i} d(x, m_i)$$

C_i = i-tes Cluster
 m_i = Centroid von C_i

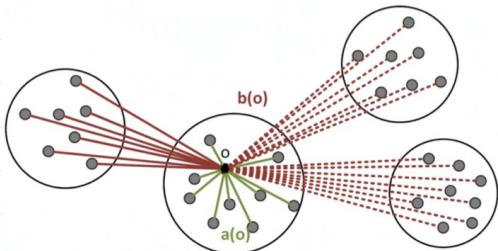
k = # Cluster, gegeben

- Minimiere E
- E wird größer falls Punkt x im "falschen" Cluster
- E ist stark abhängig von k
- Je größer k desto kleiner E
- Maximal schlechte Aussage falls k = n \rightarrow jeder Datensatz ist eigenes Cluster $\Rightarrow E = 0$

Problem: brauchen Qualitätsmaß das nicht von K abhängt

→ Silhouette-Koeffizient

- Einbeziehen von inter-/intra-cluster similarity



$a(o)$ = durchschn. Distanz zw. Objekt o und Objekten in seinem Cluster

$$A = a(o) = \frac{1}{|C(o)|} \sum_{p \in C(o)} d(p, o)$$

$b(o)$ = durchschn. Distanz zw. Objekt o und Objekten in "zweitnächstem" Cluster

$$B = b(o) = \min_{C_i \neq C(o)} \left(\frac{1}{|C_i|} \sum_{p \in C_i} d(p, o) \right)$$

Silhouette von Objekt o :

b, a gleich groß
↖

$$S(o) = \begin{cases} 0 \text{ (null)} & , \text{ if } a(o) = 0 \text{ also } |C_i|=1 \\ \frac{b(o) - a(o)}{\max\{b(o), a(o)\}} & , \text{ else} \end{cases}$$

$S(o) \in [-1, 1]$, $S(o)=1$ optimal

$S(o)$ groß wenn a klein und b groß

$s(o) = -1$: schlecht, o näher an B als an A

$s(o) = 0$: zwischen A, B

$s(o) = 1$: gute Zuordnung von o zu seinem Cluster

Silhouette-Koeff. eines Clusterings $C = (C_1, \dots, C_k)$

$$S_C(C) = \frac{1}{|C|} \sum_{C_i \in C} \underbrace{\frac{1}{|C_i|} \sum_{o \in C_i} s(o)}_{\text{Durchschn. } s(o) \text{ pro Cluster}}$$

$\underbrace{\quad}_{\text{Durchschn. über alle Cluster}}$

$0,7 < S_C \leq 1$: gute Strukturierung

$0,5 < S_C \leq 0,7$: mittelmäßige Strukturierung

$0,25 < S_C \leq 0,5$: schwache Strukturierung

$S_C \leq 0,25$: keine Strukturierung

→ weitgehend unabh. von k :

- k klein: Abstand zu m_i groß verglichen mit gr. k
- größere Abstände zu anderen Clustern

⚠ Fall $k = n$ nicht abgedeckt

Clustering-Algorithmen

- Anforderungen:
 - effektiv
 - effizient
 - Skalierbar bzgl. großer, hoch-dim. DB mit hohem noise-Anteil
 - # Datenpunkte N
 - # Dimensionen d
 - Noise Anteil

Distanzfunktionen für Objektmengen

geg. $\text{dist}(p, q)$ für $p, q = \text{Datenobjekte}$

· Single Link: $\text{dist}_{\text{se}}(X, Y) = \min_{x \in X, y \in Y} \text{dist}(x, y)$

· Complete Link: $\text{dist}_{\text{c}}(X, Y) = \max_{x \in X, y \in Y} \text{dist}(x, y)$

· Average Link: $\text{dist}_{\text{av}}(X, Y) = \frac{1}{|X||Y|} \cdot \sum_{x \in X, y \in Y} \text{dist}(x, y)$

Partitionierende Verfahren

jeder Datenpunkt wird genau einem Cluster zugewiesen

K-means

Medoid = Platzhalterpunkt für Clusterschwerpunkt

1) init: bestimme k Medoids p_1, \dots, p_k (Seeds) für geg. k, DB

2) Loop:

a) Ordne jeden Datenpunkt $\in DB$ dem nächsten Medoid zu:
minimiere Criterion Function

$$E = \sum_{i=1}^k \sum_{c \in C_i} d(c, p_i)$$

b) Berechne Schwerpunkte der Cluster neu

c) Break: Keine signifikante Verbesserung mehr

Initphase:

Ziel: finde möglichst gute Seeds \rightarrow d.h. Seeds decken Raum gut ab; Seeds \approx Cluster

Greedy - Verfahren: wähle einen Seed nach dem anderen so, dass Abstand zu bisherigen Seeds maximal

! Seed = Datenpunkt aus DB

(-) Probleme bei schlechten Seeds:

- lokales statt globales Minimum
- schlechtere Rechenzeit

k-means Varianten

• verwende nur Samples statt ges. DB

Parameter: Stichprobengröße

• mehrere Runs mit versch. Init - Seeds

Parameter: # Runs (gute Ergebnisse bei 5 runs)

• Ersetze medoid dem Raum Punkte zugewandt werden

\hookrightarrow könnte Outlier sein

$$\text{zB falls } \# \text{ Punkte} < \frac{N}{k \cdot \text{minDev}} \hookleftarrow \text{konstante}$$

(-) Nachteile k-means

- keine Garantie dass globales Optimum \rightarrow stark abh. von init
- partitionierend: jeder Datenpunkt zwingend einem Cluster zugewandt

CLARANS Erweiterung / Verbesserung k-means

Darstellung als Graph:

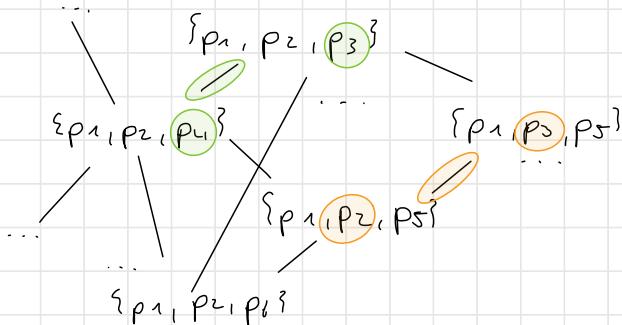
Knoten = Menge der Datenobjekte die Merkmale sind

Kanten zw. Knoten die sich in genau einem Objekt untersch.

Init: wähle einen Knoten als Seeds

→ CLARANS betrachtet pro Schritt ein Sample von Nachbar-Knoten

- vergleiche Qualitäten der Clusterings und gehe von dem besten Knoten aus weiter



Params: # betrachteter Nachbarn

Runs

Abbruchkriterium

"weniger" unkontrolliert da mehrere Nachbarn betrachtet

aber gleiche Probleme wie k-means

BIRCH

→ hierarch. Verfahren
agglomerativ

= Balanced Iterative Reducing and Clustering using Hierarchies

Aufbau Baum der Datenverteilung beschreibt

⊕ Systematischer als partitionsbasierte Verfahren

- weniger Speicherbedarf - " -

IO Kosten linear abh. mit DB Größe

eine Iteration liefert Clustering, mehrere: Verfeinerung

k ist Parameter des Algos

Kennzahlen Cluster $C = \{\vec{x}_i\}$ (x_i können auch Vektoren sein)

Centroid = Mittelpunkt

$$\vec{x}_0 = \frac{\sum_{i=1}^N \vec{x}_i}{N}$$

Radius = Ø Abstand vom Mittelpunkt

$$R = \sqrt{\frac{\sum_{i=1}^N (\vec{x}_i - \vec{x}_0)^2}{N}}$$

Durchmesser: Ø Abstand zw. allen Objektpaaren

$$D = \sqrt{\frac{\sum_{i=1}^N \sum_{j=i+1}^N (\vec{x}_i - \vec{x}_j)^2}{N(N-1)}}$$

→ groß wenn Objekte weit voneinander entfernt

Durchschn. Inter-Cluster-Distanz von C_1, C_2

$$|C_1| = N_1$$

$$i = 1, \dots, N_1$$

$$|C_2| = N_2$$

$$j = N_1 + 1, \dots, N_1 + N_2$$

$$D_2 = \sqrt{\frac{\sum_{i=1}^{N_1} \sum_{j=N_1+1}^{N_1+N_2} (\vec{x}_i - \vec{x}_j)^2}{N_1 N_2}}$$

alle Paare von Objekten mit $\vec{x}_i \in C_1$
 $\vec{x}_j \in C_2$

Clustering Feature Cluster jetzt $\hat{=}$ irgendeine Menge von Punkten

CF = (N, \vec{LS}, \vec{SS}) = aggregierte Info über Cluster

N = # Punkte im Cluster

\vec{LS} = Linear sum = $\sum_{i=1}^N \vec{x}_i$

\vec{SS} = square sum = $\sum_{i=1}^N (\vec{x}_i)^2$

} Radius, etc.
daraus berechenbar

• Cluster mergen: CF des neuen Clusters:

$$\begin{aligned} N &= N_1 + N_2 \\ \vec{CS} &= \vec{CS}_1 + \vec{CS}_2 \\ \vec{SS} &= \vec{SS}_1 + \vec{SS}_2 \end{aligned}$$

CF - Tree

- höherbalanciert
- Knoten = Cluster
- Knoten zu Cluster A ∈ Knoten zu Cluster B
→ $A \in B$
- Blatt = Menge von CF = 'Elementar-Cluster'
→ also Menge von Clustern die zusammen gehören
- innere Knoten = Eintrag [CF_i, child_i] wobei CF_i zu child_i gehört
- Baumgröße relativ unabhängig von # Datenobjekte
→ da Cluster gespeichert werden im Blatt statt einzelne Daten

Parameter:

- B' = Kapazität eines Blatts
- B = Fan-Out = Kapazität eines inneren Knoten
- T = Schwellenwert für Durchmesser / Radius eines Elementar-Clusters
→ ohne T:
 - Knoten würden nur basierend auf B' gesplittet
↳ T bringt bessere Struktur

Einfügen in den CF-Tree

- Punkt wandert in Knoten zu dessen Centroid er den kleinsten Abstand hat
- falls T verletzt wird: neuer Blatt-Cluster
- Knoten splitten falls B verletzt (d.h. zu viele Teil-Cluster)
- kein Speichern der Datenobjekte sondern modifizieren von CF

Splitting von Knoten

- neue Cluster-Seeds: am weitesten voneinander entfernte Punkte
- Zuordnung Punkte zum näheren Seed
- Knoten der einem Seed entspricht voll? anderen Knoten auffüllen
- Zum Splitten sind keine konkreten Punkte nötig \rightarrow CFs aus Ebene darunter reichen (einfach neu summieren)

$$\begin{array}{c} C = CF_1 + CF_2 + CF_3 \\ \diagdown \quad \diagup \\ C_1 \quad C_2 \quad C_3 \end{array} \xrightarrow{\text{split}} \begin{array}{c} C_a = CF_1 + CF_2 \\ \diagdown \quad \diagup \\ C_1 \quad C_2 \end{array} \quad C_b = C_3$$

\rightarrow Problem: Splitten wenn voll entspricht nicht Qualität des aktuellen Clusterings

\rightarrow zwischendurch mergen von Geschwisterknoten

- (+) spart Platz
 - bessere Clustering - Qualität

Merge im Anschluss an Splitting

- betrachte Kinder des Knotens der nach Splitting 2 neue Knoten enthält
- \rightarrow fasse die 2 Kinder mit min. Abstand zu neuem Knoten zusammen
- Erfordert ggf. Resplitting der Kinder

Clustering Algorithmus

- 1) Aufbau CF-Tree
- 2) 2 Möglichkeiten

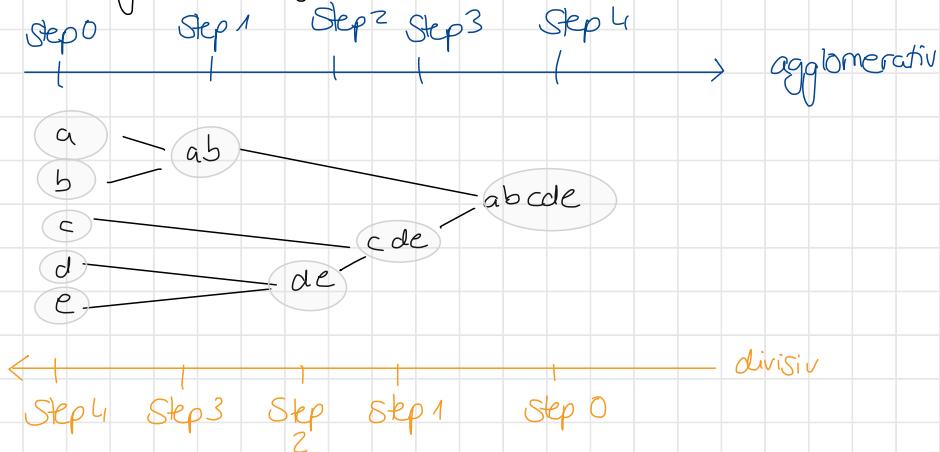
- a) Knoten des Baums = Cluster \rightarrow ggf. nicht partitionierend
 - (+) einfach: Datenobjekte werden nicht mehr betrachtet
 - (-) geringere Qualität:
 - Ausreißer 'neben' Cluster, Abstand $< T$
 - Cluster 'zu groß für T'
- Variante: K Knoten des Baums mit den meisten Datenobjekten überlappungsfreie Knoten (partitionierend)

- b) Baum verwenden um Seeds zu ermitteln
- einzelne Datenobjekte anschauen
 - z.B. Mittelpunkte der k größten Elementarcluster als Seeds für z.B. k-means

Hierarchisches Clustering

- globale Params für partit. Clustering - Verfahren zur Identifizierung aller Cluster gibt es i.a. nicht
- hierarchisches zerlegen des DB in geschachtelte Cluster

Darstellung: Dendrogramm



- Wurzel = gesamter DB
- Blatt = einzelnes Datenobjekt
- innerer Knoten = Vereinigung der Datenobjekte seiner Teilbäume
- Höhe innerer Knoten = Abstand zw. seinen 2 Kindknoten

Agglomerativ bottom-up

Init: Jedes Objekt einem Cluster

Loop:

- berechne paarweise Abstände zw. Clustern

- merge Paar $\{A, B\}$ mit kleinstem Abstand zu $C = A \cup B$

- entferne A, B aus Menge der Cluster

- Füge C ein

Break: Aktuelle Clustermenge besteht nur noch aus C

(Optimierung: nur noch Distanz von C zu restl. Clustern berechnen, rest hat sich ja nicht geändert)

Komplexität: $O(n^3)$ (n^2 Möglichkeiten für merge pro Schritt)

Divisiv top-down

Init: ein Cluster mit allen Datenobjekten

Loop:

- break wenn alle Objekte eigenes Cluster

- wähle Split-Cluster

- Ersetze durch sub-Cluster

Beispielverfahren: DIANA

- wähle Cluster mit größtem Durchmesser
- Suche nach abseitigstem Objekt $o \in C$ (höchster Ø Abstand zu anderen Objekten $\in C$)

→ • Splinter-Group $\{o\}$
 $= SG$

- wdh Einfügen von $o' \in C \setminus SG$ mit größtem $D(o') > o$ in SG
 bis für alle $o' \in C \setminus SG$ gilt: $D(o') \leq o$

$$D(o') = \sum_{o_j \in C \setminus SG} \frac{d(o', o_j)}{|C \setminus SG|} - \underbrace{\sum_{o_i \in C \setminus SG} \frac{d(o', o_i)}{|SG|}}_{\emptyset \text{ Abst. zur SG}}$$

$\emptyset \text{ Abst. zu Rest}$

$\rightsquigarrow D(o') \leq o$: o' gehört eher zu Rest als zu SG

Diskussion

agglomerativ

1. Schritt: $n(n-1)/2$ Kombis

- benötigt $n-1$ Schritte

divisiv

1. Schritt: $2^{n-1}-2$ Möglichkeiten für DB-Split (DIANA: nicht alle explizit betrachtet)

- Berücksichtigt lokale Muster

- konzeptionell anspruchsvoller (Vorgehen bei Split nicht offensichtl.)

- berücksichtigt globale Datenverteilung
 → evtl. bessere Resultate

Hochdimensionale Merkmalsräume

- Interessante Cluster i.d.R. nicht in allen Dim. Cluster
→ bisherige Ansätze funktionieren nicht
- zu geringe Dicht mit dichte-basierten Ansätzen
 - keine sinnvolle Struktur der Daten mit hierarch. Verfahren

Problem:

- finde Dim. in denen Punkte Cluster bilden
- ohne Domänenwissen schwierig

→ Unterscheidung subspace clustering
projected clustering → einfacher, da Objekt nur zu 1 Cluster gehören kann

Projected Clustering

Input: $k = \#$ Cluster die zu finden sind
 $\ell = \#$ Dim. pro Cluster

Output: Partitionierung der Daten in $k+1$ Mengen
→ noise in Cluster C_{k+1}
+ Infos in welchen Dim. D_i Cluster C_i ist

Algorithmus ($\approx k$ -means)

Init: wähle Seeds

Loop:

- 1) bestimme Dimensionen zu jedem Medoid
- 2) ordne jedes Datenobjekt dem nächsten Medoid zu
- 3) berechne Schwerpunkte neu

break: keine signifikanten Änderungen mehr

Dimensionen ermitteln

• ggg. Menge von Medoiden \rightarrow wie Dimensionen wählen?

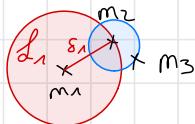
\rightarrow Betrachte Nachbarschaft L_i von m_i :

$$\delta_i = \min_{i \neq j} (\text{dist}(m_i, m_j))$$

\rightsquigarrow Suche nächsten Medoid von Medoid m_i

δ_i ist Radius einer Kugel um m_i

$\rightarrow L_i$ sind Punkte innerhalb der Kugel



L_i kann auch leer sein

\rightarrow Berechne Ø-Distanz der Punkte in L_i pro Dimension j :

\rightarrow also z.B. Ø-Distanz der Punkte bzgl. x-Achse, ... $x_{i,j}$

\rightarrow Wähle die Dimensionen für die die Punkte nahe bei m_i sind

\rightarrow Mittel der Ø-Distanzen über die Dimensionen

$$i = \text{Clusternummer} \quad y_i = \frac{\sum_{j=1}^d x_{i,j}}{d}$$

Ist $x_{i,j} - y_i < 0$: Dim. j tendenziell wichtig für Medoid;
 $<< 0$: sehr wichtig

Interessant ist relative Abweichung vom Durchschnitt:

$$\text{Stdev } \sigma_i = \sqrt{\frac{\sum_j (x_{i,j} - y_i)^2}{d-1}}$$

(Ø Abweichung vom Mittelwert)

→ Vergleiche Differenz für untersch. Cluster

$$z_{i,j} = \frac{k_{i,j} - y_i}{\sigma_i} \quad \sigma \text{ klein} \rightarrow z \text{ groß}$$

~ Sortiere $z_{i,j}$, jedem Cluster Dimensionen zuordnen

→ Bestimmung der Cluster:

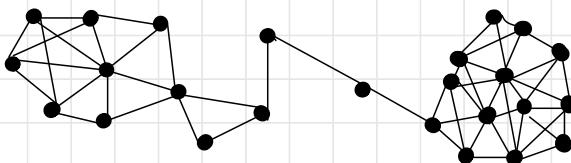
Für jeden Punkt Manhattan Segmental Distanz zu jedem Medoid

Clustering mit kategorischen Attributen

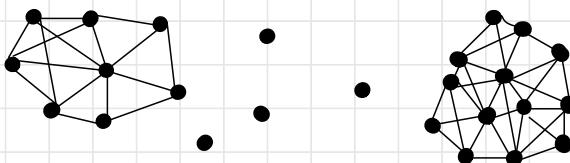
zB Lieblingsfarbe, boolsche Werte, mengenwerte, ...

Link-basierte Methoden

Verbinde alle Punkte mit Distanz $< d$ zu Graph



~ So noch nicht hilfreich → Minimalzahl von Nachbar-Punkten: 3



· unformige, langzogene, konkav Cluster

· untersch. große Cluster (wenn Dichte nicht unter Schwellenwert fällt)

→ Frage: was wird ausgegeben? Mittelpunkte? → geht schlecht bei

→ abh. von Anwendungsfällen zB Kugeloberfläche

Kategorische Attribute

• Allg.: Wertebereich ist endliche Menge ohne natürliche Abstandsmaß

z.B. Farben $\rightsquigarrow d(\text{grün}, \text{rot})?$

Bisherige Verfahren erfordern Distanz \rightarrow gut für numerische Attribute

problem: hohe Dim. \rightarrow trotz ähnlicher z.B. Symptome wenig gemeinsam

z.B. Featurevektor / Bauchweh 1, Bauchweh 2, Husten 1, Husten 2)

Pat1 : (1, 0, 1, 0)

Pat2 : (0, 1, 0, 1)

Abstandsmaß 1: similarity $(x_i, y_i) = 1$ falls $x_i = y_i$
0 sonst

\rightsquigarrow Pat1, Pat2 demnach unähnlich

\rightsquigarrow fasst viele Pat. zusammen die viele Symptome beide nicht haben \rightarrow nicht Ziel der Sache

Jaccard-Koeffizient

Maß der Überlappung zweier Mengen

$$\frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

Größe des Schnitts
Größe der Vereinigung

Wertebereich: [0, 1]

0 falls Schnitt leer
1 falls gleiche Mengen

⊖ Lässt Nachbarschaft außer Acht \rightarrow vergleicht nur Punkte Transaktionen

zB. Patienten im Winter

$\{1, 2, 3\}$
 $\{1, 2, 6\}$

$\{1, 2, 4\}$
 $\{1, 7\}$

$\{1, 2, 5\}$
 $\{2, 6\}$

$\{1, 5\}$

C_1
 C_2

$1, 2 =$ Erkältung

$1 =$ Husten

$2 =$ Schnupfen

$3, 4, 5 =$ Kopfschmerzen

$6, 7 =$ Augenschmerzen

→ $1, 2$ nicht diskriminierend → nicht sinnvoll als Split-Attribut für Clustering

→ Appl. Verfahren könnte zuerst $\{1, 2, 3\}$ und $\{1, 2, 6\}$ zusammenfassen ↗

Lösung: Links

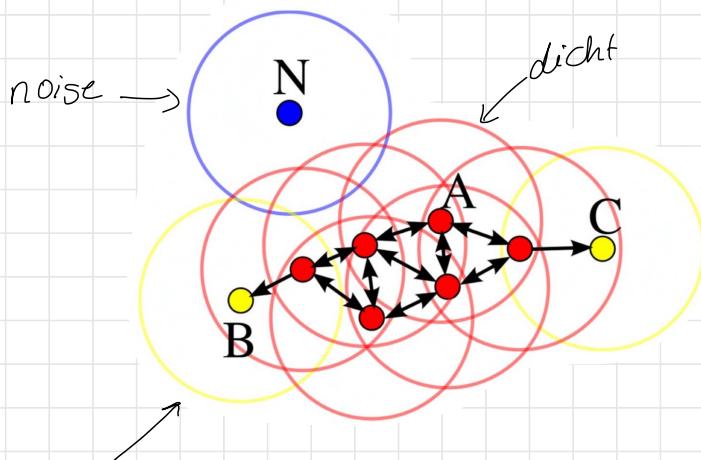
- 2 Punkte = Nachbarn wenn Ähnlichkeit $> \Theta$ (Schwellenwert)
zB Transaktionen, Ähnlichkeitsmaß: Jacc. Koeff.
- Anzahl der Links zw. 2 Punkten =
Anzahl gemeinsamer Nachbarn
- Clustering mit diesem Ähnlichkeitsmaß
- Unabh. von Verfahren und Ähnlichkeitsmaß in Schritt 1
zB k-means + Jacc. K.

Dichte-basierte Verfahren

• Dichte = Anzahl was auch immer pro Volumeneinheit

DBSCAN

• Objekte dicht ("core object") = mind. minPts andere Objekte in Kugel ums Objekt mit Radius ϵ



= liegt in ϵ -Umgebung eines dichten Objekts und ist selbst nicht dicht (Cluster-Rand)

dichte-verbunden = dicht oder dichte-erreichbar

• Zuordnung dichter Objekte deterministisch
dichte-erreichbarer nicht \rightarrow abhängig von Init
welchem Cluster zugewandt

Algorithmus DBSCAN

Loop: für alle unbesuchten Punkte $p \in DB$:

- 1) markiere p als besucht
- 2) berechne N : alle n in ϵ -Umgebung von p
 - a) falls $\text{size}(N) < \text{minPts}$
→ p nicht dicht
 p als Noise markieren
 - b) p dicht:
 - neues Cluster C mit p aufmachen

Loop: für alle $n \in N$:

- a) falls n noch keinem Cluster zugeordnet:
 $\text{recursive Expand Cluster}(n, C, \text{minPts}, \epsilon)$

$\text{recursive Expand Cluster}(n, C, \text{minPts}, \epsilon)$

- 1) Füge n zu C hinzu
- 2) Falls n noch nicht besucht:
 - markiere n als besucht
 - berechne N alle n' in ϵ -Umgebung von n

Falls $\text{size}(N) \geq \text{minPts}$:

- Loop: für alle $n' \in N$:
- falls n' noch keinem Cluster zugeordnet:
 $\text{recursive Expand Cluster}(n', C, \text{minPts}, \epsilon)$

Varianten

- ohne Rekursion: speichere zu expandierende Objekte in Liste
= ControlList

Komplexität

$O(n \cdot \text{Laufzeit}(\varepsilon\text{-Nachbarschaftsanfrage}))$

- ohne räumlichen Index worst case $O(n^2)$
(alle Elemente des DB in ε -Umgebung)
- mit räuml. Indexstruktur (zB R-Baum) $O(n \log n)$

Parameter

- kleineres minPts: größere Cluster
- kleineres ε : mehr Cluster, kleinere Cluster, manche Cluster verschwinden
 \rightarrow rote Kugeln werden gelb oder blau

OPTICS = Ordering Points To Identify the Clustering Structure

Output: Profil des Clusterings

\rightarrow keine konkreten Cluster

\rightarrow Infos über "wie viele Cluster gibt es unter welchen Umständen im Datenbestand"

Problem DBSCAN: ε gut wählen

\rightarrow jetzt: Wähle nur ε_{\max} , untersuche alle $\varepsilon \leq \varepsilon_{\max}$ minPts immer noch fix

\rightarrow generiere Datenstruktur aus der sich Clustering für Wert ε ablesen lässt

→ Diagramm:

x = cluster-ordering

- Objekte so hintereinander angeordnet,
dass nahe Objekte hintereinander

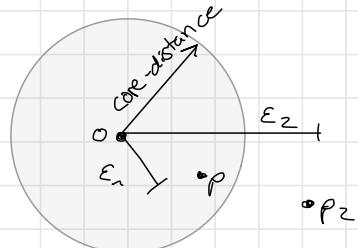
y = reachability distance

core-distance

core-distance_{minpts}(o) = kleinster Wert den ϵ annehmen kann sodass o dichtes Objekt

reachability distance

= kleinster Abstand von o zu bereits eingefügtem Objekt



- ist p näher an o als core-dist: genaue dist. egal
- $\epsilon < \text{core-dist} \rightarrow o$ nicht dicht
- $\epsilon > \text{core-dist} \rightarrow p$ ist von o dichte-erreichbar
- p weiter weg als core-dist: ob p von o dichte-err. abhängig von ϵ

$$\text{ReachDist}_{\epsilon_{\max}, \text{minpts}}(p, o) = \begin{cases} \text{dist}(p, o) & \text{dist}(p, o) > \text{core}(o) \\ \text{core}(o) & \text{dist}(p, o) \leq \text{core}(o) \\ \text{undefined} & \text{dist}(p, o) > \epsilon_{\max} \end{cases}$$

$$\begin{aligned} & \text{dist}(p, o) > \text{core}(o) \\ & \text{dist}(p, o) \leq \text{core}(o) \\ & \text{dist}(p, o) > \epsilon_{\max} \end{aligned}$$

! nicht symmetrisch $\text{reachDist}(p, o) \neq \text{reachDist}(o, p)$

Algorithmus

Output: ordered-list

Loop: für alle $n \in$ Datenbestand:

falls Objekt noch nicht bearbeitet:

1) markiere als bearbeitet

2) $\text{reach.dist}(n) = \text{undefined}$

3) berechne N : Nachbarpunkte in E_{\max} -Umgebung

4) berechne $\text{coreDist}(n)$

5) Füge n zur ordered-list hinzu

Falls $\text{coreDist}(n) \neq \text{undefined}$:

1) ControlList.update(N, n)

Loop: While ControlList.notEmpty :

1) $c = \text{ControlList.next}()$

2) markiere c als bearbeitet

3) berechne C : Nachbarpunkte in E_{\max} -Umgebung

4) berechne $\text{coreDist}(c)$

5) Füge c zur ordered-list hinzu

Falls $\text{coreDist}(c) \neq \text{undefined}$:

1) ControlList.update(C, c)

ControlList.update(N : neighbors, n : centerObject):

1) $cd = \text{core-dist}(n)$

Loop: f $p \in N$: falls p noch nicht bearbeitet:

1) new-reach.dist = $\max(cd, \text{dist}(n, p))$

Falls $p.\text{reach.dist} = \text{undefined}$:

// p not in ControlList

1) $p.\text{reach.dist} = \text{new-reach.dist}$

2) ControlList.insert($p, \text{new-reach.dist}$)

Ansonsten: // p is in ControlList, check for improvement

Falls $\text{new-reach.dist} < \text{reachDist}(p)$:

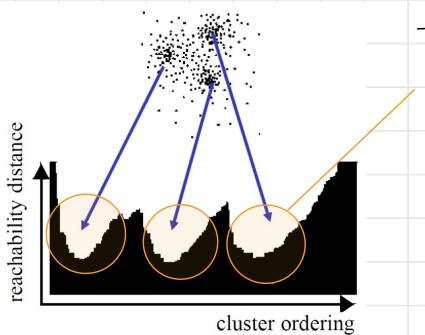
$\text{reachDist}(p) = \text{new-reach.dist}$

ControlList.moveUp($p, \text{new-reach.dist}$)

ControllList

- Priority Queue
- Sortierkriterium: reach_dist zu bereits ausgegebenen Objekten
- nur Objekte die noch nicht in output-list
- muss leer sein bevor neues zufälliges Objekt aus DB geholt wird

Output



Täler \leq Cluster
→ nahe Objekte hintereinander,
Höhe der Balken $\hat{=}$ reach_dist .
Objekte des gleichen Clusters folgen aufeinander

Performanz

\approx DBSCAN

- O(n · Laufzeit (ϵ - Nachbarschaftsberechnung))
 - ohne räuml. Indexstrukturierung worst case $O(n^2)$
 - mit average case $O(n \log n)$
 $\rightsquigarrow O(n^2)$ falls alle Objekte aus DB in Nachbarschaft

Parameter

- ziemlich insensitiv
- gute Resultate solange Params genügend groß
- ϵ_{\max} kleiner: große Nachbarschaften (Cluster mit geringer Dichte, größerer Fläche) außen vor
- ϵ_{\max} zu groß: alle Punkte aus DB sind in Nachbarschaft
- min pts kleiner: zu kleinteilige Cluster \rightsquigarrow Objekte mit wenig Nachbarn zählen schon als dicht \rightarrow Cluster

Cluster - Ensembles

Ziel: robuste Clustering - Ergebnisse
→ kombiniere Resultate versch. Clustering - Modelle

Unterscheidung:

- Modell-basiert: z.B.
 - versch. Verfahren
 - ein Verfahren versch. Parametrisierung / Initialisierung
 - Verfahren die mehrere Lösungen liefern
- Daten-basiert:
 - versch. Teilmengen der Daten
 - Beschränkung auf versch. Dimensionen

Verfahren

- generiere K versch. Clustering - Ergebnisse
- erstelle Graph (s.u.)
- Graph - Partitionierungs - Algo anwenden:
 - Partitionen $\hat{=}$ Clustern (evtl. überlappend!)
 - Ergebnisse der versch. Verfahren sind berücksichtigt

Hypergraph

- Knoten = Datenobjekte
- Kanten = Objekte verbunden falls gleiches Cluster gemäß einem Verfahren, Gewicht gibt an wie viele Verfahren so zugewandt haben
 - ↪ Hyper-Kanten: Kanten zwischen n Objekten (statt $n=2$)
 $\hat{=}$ Clustern der versch. Verfahren
- Constraints, z.B. für Balanciertheit bei Bedarf

Meta-Clustering

- Knoten = Cluster
- Kanten bei Überlappung der Mengen
- Gewicht = Jaccard-Koeffizient

Probabilistisches Clustering

- Clusterbeschreibung verteilungs basiert
→ für Beschreibung eines Clusters ist Varianz, Mittelwert ausreichend
 - oft keine genaue Zuordnung möglich → Wahrscheinlichkeitsbasierte Zuordnung
- Also: Finde Cluster die am besten zu gpg. Daten und Apriori Erwartungen passt

Finite - Mixture Problem

- Wahrscheinlichkeitsverteilung beschreiben durch k Wkt. Verteilungen
 - jede der k Verteilungen könnte Cluster beschreiben
 - Verteilung im Clustering nicht bekannt
- Cluster selbst versch. Wahrscheinlich
z.B. 10mal mehr Hustenpatienten als Bauchwehpatienten

Clustering mit Finite-Mixtures

geg.: Datenobjekte, Anzahl + Art der Verteilung der Cluster

→ Problem: finden der Params

- ohne jegliche Annahme über Verteilung i.A. nicht möglich

Parameter zur Cluster-Beschreibung

x_1, x_2, \dots, x_{n_A} = Datenobjekte aus Cluster A (insg. n_A)

$n = \#$ Datenobjekte gesamt

$$p_A = \frac{n_A}{n}$$

$$\sigma_A^2 = \frac{1}{n_A - 1} \sum_i (x_i - \mu_A)^2$$

$$\mu_A = \frac{1}{n_A} \sum_i x_i$$

Expectation Maximization iteratives Vorgehen

Beispiel für $k=2$

1) Initialisierung der Parameter $(p_A, \mu_A, \mu_B, \sigma_A^2, \sigma_B^2)$ raten

2) Für alle $x \in DB$ Cluster-Wkt. berechnen (Expectation-Schritt)

$\Pr[A|x]$ bzw. $\Pr[B|x]$ ↗ Dichtefkt.

$$\Pr[A|x] = \frac{\Pr[x|A] \cdot \Pr[A]}{\Pr[x]} = \frac{f(x; \mu_A, \sigma_A^2) \cdot p_A}{\Pr[x]}$$

(da $\Pr[A|x] + \Pr[B|x] = 1$ Nenner ergl.)

3) Parameter der Cluster neu berechnen * (Maximization-Schritt)

4) Wdh bis Abbruchkrit. erreicht

* abhängig von Wkt dass Objekt x_i zu Cluster A gehört

$$w_{i,A} = \Pr[A|x_i]$$

$$\mu_A = \frac{\sum_i w_{i,A} x_i}{\sum_i w_i}$$

$$\sigma_A^2 = \frac{\sum_i w_{i,A} (x_i - \mu_A)^2}{\sum_i w_{i,A}}$$

Abruchkriterium

- Fixpunkt wird nie ganz erreicht
 - Berechne Clustering - Gütemaß nach jeder Iteration
 - "Overall likelihood" $\prod_i (p_A \cdot \Pr[x; |A] + p_B \cdot \Pr[x; |B])$
 - ~ Verteilungen beschreiben DB gut: viele Instanzen nahe bei einem Mittelpunkt $\rightarrow \Pr[x; |A|B]$ groß
 - ~ schlecht: beide \Pr_s klein
 - Prüfe ob sich Güte verbessert hat seit letztem Check
- ② Findet nur lokales min
- Verfahren mit versch. Parametrisierungen
 - wähle bestes Clustering-Ergebnis

Erweiterungen Mixture Model

- > 2 Klassen
- > 1 Attribut
- Korrelationen zw. Attributen] auch
- Kategoriale Attribute] Kombis davon
- Andere Verteilungen als Normalverteilung

mehr als 1 Attribut, Korrelationen

- A. reichen Kovarianzen nicht aus
- Maß für Korrelation: Kovarianz
- Kovarianzmatrix (symmetr.): Darstellung Korrelationen zw. Attributen

einfach falls Attribute unabhängig:

- alle Kovarianzen = 0 \rightarrow müssen nicht betrachtet werden

$$\sim \text{Modell wird zu } \Pr[A|x,y] = \frac{\Pr[x|A] \Pr[y|A] \cdot \Pr[A]}{\Pr[x,y]}$$

- Korrelierte Attribute: Ellipse mit Breite σ_x , Höhe σ_y
- ↪ Punkte auf Ellipse = gleiche Wkt.
- 3 Params: σ_x , σ_y , $\text{cov}(x,y)$ für Ausrichtung der Ellipse

- # Params bei n Attributen:
 - alle unabh.: $2n (\sigma, \mu)$

Kovariant: $n + \frac{n(n-1)}{2}$ (n mal μ , $n \times n$ für Kov. matrix)

Kategorische Attribute → x kategorisch mit v möglichen Ausprägungen
 $\Pr[A] = p_A$ → keine Änderung

$\Pr[x|A]$ ist jetzt Zähldichte statt Dichtefkt.

Bei k Klassen: $k \cdot v$ Parameter (zB Wkten der einzelnen Ausprägungen)

Änderung Algo:

1, 2 wie oben

3) Maximization: Berechnung der Ausprägungswkts aus vorhandenen Instanzen und Wkts. ihrer Klassenzugehörigkeit.

→ deutlich mehr Modellparams

→ Auswirkung auf Qualität der Schätzung, Overfitting

Software für EM-Algo

Eingabe: # Cluster, Attributtypen (num/kat.), Korrelationen, Umgang mit null-Werten

Verteilungen:

- Normalvert. nicht immer gut
- Attribute mit min, ohne max → log-normale Vert.
- Integer-Attribute: Poisson-Vert.

Overfitting

durch:

- zu viele Params
→ nicht alle Attribute als Kov. markieren
 - # Cluster k zu hoch gewählt; Extremfall $k = n$
 - Verteilungen mit zu vielen Params
- zu viele Params 'bestrafen': Laplace-Estimator
- verlange dass kat. Attributwerte mind. einmal als $w_{kt} > 0$ markiert
 - Falls ein Cluster eig. leer ist wird Gütemaß wegen Laplace Estimator schlechter

Outlier

Definition 1 (intuitiv): Objekt des DB, das in bestimmter Hinsicht erheblich vom Rest des DB abweicht

Anwendung

grundätzliche Unterscheidung:

1) inhaltliche Anwendung bei "guten" Daten
z.B. Fraud Detection, Videoüberwachung

2) Data Cleaning: Verbessere Qualität der Daten

Ansätze

- 1) Verteilungsbasiert
- 2) Clustering basiert
- 3) Abstandsbasiert
- 4) Dichtebasiert

Verteilungsbasiert

geg.: Verteilung des Datenbestands
zB Normalverteilung

Test auf Outlier basierend auf:

- Wkt des Vorkommens des betrachteten Datums
- Datenverteilung
- Parameter der Verteilung (Mittelwert, Varianz,...)
- # der erwarteten Ausreißer

Anwendung: Datenbereinigung vor der eigentlichen Analyse



- meisten statistischen Tests nur für ein Attribut
- Datenverteilung oft unbekannt
- statistische Maße ggf. abhängig von Outliern
zB Mittelwert
→ ggf. nicht alle / falsche Outlier detektiert

Clustering-basiert

Outlier als Nebenprodukt des Clustering

Ausgangspunkt: Clustering-Ergebnis

→ betrachtet Cluster-Randpunkte,
Punkte ohne Zuordnung, ...

Abstands-basiert

Definition 2: Objekt O aus Datenbest. T ist $\text{DB}(p, D)$ -Outlier wenn Abstand von O zu mind. p -Prozent der Daten aus T größer als D ist.

- ⊖ Abhängig von Parametern $p, D \rightarrow D$ sehr unintuitiv

Alternative: k-NN

- Abstand basiert jetzt auf Distanz zum k -nächsten Nachbarn
- Berechne für alle Punkte den Abstand zu seinem k -NN
- Sortiere Punkte absteigend nach dieser Distanz
- ersten Elemente der Liste sind Outlier
- wie viele davon wirklich Outlier: Nutzer kann flexibel entscheiden

- ⊕ · flexibler
- nur noch ein Parameter k ($\hat{=} p$)
- intuitiver als ①

- ⊖ · Wert von k entscheidend
 - zu klein: mehrere Outlier die nahe beieinander liegen werden nicht erkannt (Extremfall: 2 Objekte bei $k=1$)
 - zu groß: zu viele Punkte werden als Outlier erkannt
↳ Problem bei kleinen Clustern

Algorithmen

Index-basiert

- k -Abstand = Abstand zum k -nächsten Nachbarn
- k -Abstand $< D \rightarrow$ kein Outlier
- schaue ob Kugel mit Radius D um Punkt genug Punkte enthält
- Bereichsanfrage an zB R-Baum

-

- muss für alle Datenpunkte extra gemacht werden
↳ kein "find all outliers"
- IMPROVEMENT: Algo zur Bereichsanfrage vorzeitig abbrechen falls genug Objekt in Nachbarschaft
- teuer, falls Index erst aufgebaut werden muss

Nested - Loop

eigentlich: join zweier Relationen (DB)

JETZT: Self-join also 1.Relation = 2.Relation = Datenbestand
⇒ nested-loop Algo damit;
statt Attributvergleich: Abstand berechnen

Loop 1: gehe über alle $p \in$ Relation 1:

Loop 2: gehe über alle $q \in$ Relation 2:

- berechne $d = \text{dist}(p, q)$

- falls $d < D \rightarrow \text{count}(p)++$

Falls count höher als erforderliche # Werte $\rightarrow p$ kein Outlier
↳ frühzeitig abbrechen spart Zeit!

Komplexität $O(n^2)$

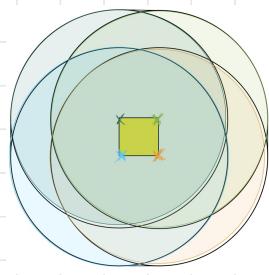
Verbesserung: Betrachte mehrere Objekte $p_1, \dots, p_k \in$ Relation 1 auf einmal als Batch
→ Wähle Größe der Batches \approx Hauptspeichergröße
↳ immer noch $O(n^2)$, aber absolut weniger Iterationen

⊕ einfach zu implementieren

zellen - basiert

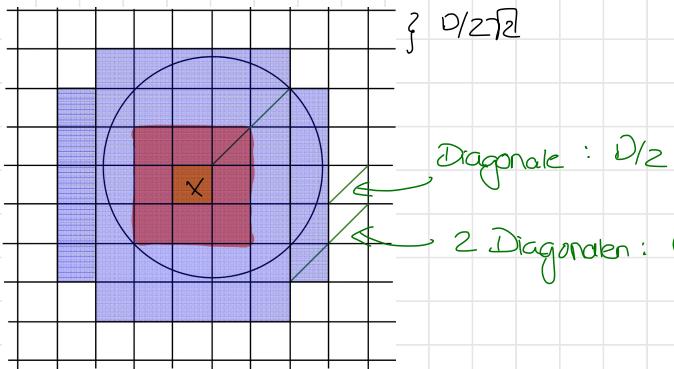
Minkowski - Summe eines Quadrats

= # Elemente die von einem beliebigen Punkt im Quadrat Abstand $\leq D$ hat



→ Partitioniere k -Dim Raum in würfelförmige Zellen der Länge $D/2\sqrt{k}$

$$k = 2$$



Betrachte alle Punkte aus Zelle x aufeinmal:

L_1 - Nachbarschaft: Zellen die x umgeben

↪ wenn in $L_1 + x \geq p \cdot N$ Objekte liegen: alle Punkte in x keine Outlier

↪ ist typischerweise oft erfolgreich (da A nicht so viele Outlier)

L_2 - Nachbarschaft: Minkowski - Kreise um x → alle Zellen die diese Kreise beinhalten

↪ wenn in $L_2 + x < (1-p) \cdot N$ Objekte: alle Punkte in x Outlier

Falls keins von beidem: Tupel einzeln inspizieren

↪ 100 - intensiv :

↪ unsichere Tupel Seitenweise in main mem. laden ↪

Dichte basiert

Idee: betrachte lokale Distanz statt globaler Distanz
 ↳ Umgebung wird beachtet
 (Bsp. Bewohner USA → Alaska vs. Star in Villa)

→ relative Distanz zu k-nächsten-Nachbarn

local-reachability-density (lrd)

Berechne Dichte von Objekt p:

Kehrwert des Ø - Abstands zu k-NN N_i ($i = 1, \dots, k$)

$$\text{also } lrd(p) = \frac{1}{\sum_{i=1}^k \text{dist}(p, N_i)}$$

je kleiner $lrd(p)$ desto weiter weg ist p von den k-NN

local-outlier-factor (lof)

$$lof(p) = \frac{\text{Ø-Dichte der k-NN}}{\text{eigene Dichte}} = \frac{\sum_{i=1}^k lrd(O_i)}{k \cdot lrd(p)}$$

wobei O_i ($i = 1, \dots, k$) = k-NN

→ je größer $lof(p)$ desto eher Outlier

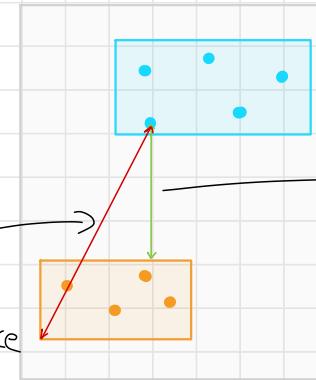
~ Verhältnis Dichte der Umgebung zur eigenen Dichte

Algorithmen Indexbasiert

ggf. räuml. Indexstruktur (zB R-Baum)

↪ berechne obere, untere Schranke für lof

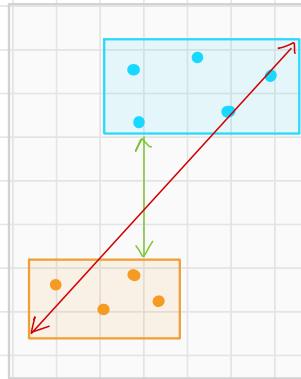
Punktbasiert:



Distanz Punkt,
weitest entferntes
Eck vom Rechteck
ist obere Schranke

Distanz (Punkt, Rechteck)
ist untere Schranke

Batch-basiert:



→ statt direkt korrekte Dicke berechnen
↪ Approximieren und ggf. frühzeitig abbrechen

Hochdimensionale Räume

Anomalien:

- Sparsity: Raum nur dünn mit Punkten besetzt
- hierarchische Datenstrukturen nicht effektiv

→ neue Outlier def:

Punkt ist Outlier wenn er in niedrigdim. Teilraum in Region mit ungewöhnlich niedriger Dichte liegt

Ungewöhnlich niedrige Dichte

Teilt jeden Wertebereich eines Attributs (jede Dimension) in ϕ Partitionen mit gleicher Tiefe (= Anzahl Objekte)

$$f = 1/\phi$$

$\leadsto \phi^d$ Partitionen

- Falls Dimensionen statistisch unabhängig
Anzahl Datenobjekte je Partition: $N \cdot f^d$ $N = \# \text{ Objekte im DB}$
- $n(D)$ = tatsächliche $\#$ Objekte in Partition D
- Idee: ungewöhnl. niedr. Dichte falls $n(D) < N \cdot f^d$
 \leadsto absolute Zahlen blöd

• Standardabweichung (Bernoulli: Punkt in Zelle ja/nein $\leadsto p=f^d$)

$$\sqrt{N \cdot p \cdot (1-p)} \approx \sqrt{N \cdot f^d \cdot (1-f^d)}$$

\leadsto Sparsity coefficient $S(D) = \frac{n(D) - N \cdot f^d}{\sqrt{N \cdot f^d \cdot (1-f^d)}}$

$S(D) < 0$: geringe Dichte

- (-) # Zellen: ϕ^d \leadsto alle Zellen betrachten zu großer overhead
 \rightarrow Cest Dim. problem nicht \rightsquigarrow

Subspace Search

Wähle nur geeignete Teiräume zur Betrachtung
↪ nicht trivial

Outlier Typen

Frage 1: Was sind Outlier bei bestimmter Kombi von Dimensionen

Frage 2: In welcher Kombi von Dims. ist Objekt Outlier

nicht-triviale Outlier

P ist nicht-trivialer Outlier in Attributraum A_p , falls

P kein Outlier in Teilraum $B \subset A_p$

↪ A_p ist also kleinste Attributmenge um P als Outlier zu rechtfertigen

Strong Outlier

P ist strong outlier in A_p , falls kein Outlier in irgendeinem $B \subset A_p$ existiert

Weak Outlier

nicht-trivial, aber nicht strong

strong \Rightarrow nicht-trivial

Weitere Ansätze zur Outlier-Detektion

Informationstheoretisch

- erstelle Modelle der Objekte → definiere wie ausführlich es sein muss
 - ↪ Outlier sind Objekte mit erheblich größerem Modell
 - ↪ outlier score = Unterschied der Ausführlichkeit des Modells

Probabilistisch

- erstelle Mixture Modell, nicht dazu passende Objekte = Outlier

Binäre, mengenwertige Daten

suche ungewöhnliche Transaktionen

- ↪ mengenwertig: ! viele Werte kommen oft nicht vor ↗ uninteressant

Ausgangspunkt: Frequent Pattern Mining

- ↪ Transaktionen die wenige Frequent Patterns enthalten sind ungewöhnlich

outlier score = Summe der support-Werte der Frequent Patterns die in Transaktion enthalten

↗ oft schlechte Unterscheidung zw. outlier, noise

Ensembles

Problem: wie vereint man mehrere Modelle / Klassifier?

Ensemble = Menge solcher Modelle

Nachteil einzelnes Modelle alleine:

Overfitting \rightarrow Classifier zu sehr auf Trainingsdaten zugeschnitten

Lösung 1: Entscheidungsbäume

"Zurückschneiden" zu starker Verästelungen

\hookrightarrow Allgemeine Lösung?

Wahrscheinlichkeiten, Conditional Risk

Erinnerung: manche Classifier liefern wkt. für Label mit zurück

\hookrightarrow Conditional Risk: berechne Kosten einer Klassifizierung

$$R(i|x) = \sum_j P(j|x) \cdot \text{cost}(i,j)$$

i = vorhersage j = ground truth

Relabeling

berechne für jedes Objekt \in Trainingsdaten die cond. risk

\hookrightarrow berechne optimale Vorhersage für jedes Objekt

z.B. verursacht geringste Kosten

und trainiere mit diesen Labels

Gewichtete Objekte

Ziel: Classifier soll sich auf hohe Gewichte konzentrieren

\rightarrow mit Resampling: ungewichteten DB aus gewichtetem machen

\rightarrow Sampling mit Zurücklegen

\rightarrow Neuer DB hat gleiche Größe wie AusgangsDB aber Gewichte sind berücksichtigt

Ansätze

- sowohl für Klassifikation als auch numerische Vorhersagen

→ Grundlage Bagging, Boosting:

- Klassifikation: Abstimmung (mit Gewichten)
- numerische Vorhersagen: (gewichteter) Durchschnitt
- hier: alle Ensemblemitglieder verwenden gleiches Modell
z.B. Entscheidungsbäume

Bagging

Erzeugt versch. Modelle durch Verwendung versch. Trainingsdaten

Ausgangspunkt: TrainingsDB der Größe N

- ↳ Samples gleicher Größe N: Ziehen mit Zurücklegen
- ↳ auf allen Samples (parallel) trainieren

Klassifizierung:

- jedes Modell liefert Vorhersage
 - häufigster Vorhersagewert gewinnt
- alle Ensemblemitglieder gleich gewichtet (haben gleich viel zu sagen)

Randomisierungstechniken

1) "Standard VTechnik" S.O.

2) für Entscheidungsbäume:

- bisher: wähle Split mit größter Entropie
- jetzt: wähle Split aus N besten zufällig
 - ~ N zu groß: Modell wird schlechter
 - ~ Modelle weichen nicht stark voneinander ab

- 3) für NN - Classifier: (zB instanzbasiertes Lernen)
- bisher: Vorhersage ist häufigster Wert in k-NN Sphere
- jetzt: Modelle in versch. Teilräumen erstellen
(also Attribute weglassen \uparrow)
- ④ Verfahren muss nicht angepasst werden

Wahrscheinlichkeiten

al Wkt ist Mittel der vorhergesagten Wkts der Modelle

b) Wkt ist relative Häufigkeit mit der Label vorhergesagt

- ⊕ · effektiv gegen Overfitting, da DB variiert
↳ leicht versch. Classifier, lokale Besonderheiten werden ausgeglichen
- bessere Wkt der Label-Vorhersagen: Fehler in Wkt werden ausgeglichen

Meta Cost

· Berücksichtigung der Kosten beim Lernen \rightarrow Relabeling + Bagging

Boosting

· alle Modelle vom gleichen Typ

Modellerzeugung: ein Modell nach dem anderen basierend auf
1) alle Objekte \in DB haben gleiches Gewicht Verringern

Loop: t mal:

- Lerne Modell auf gewichtetem DB, speichere Modell
- Berechne Fehler e des Modells auf gewichtetem DB
 $e = 0$ or $e \geq 0.5$ (irgendwas geht sehr schnell \rightarrow break)

Loop: für alle Objekte \in DB:

- falls korrekt klassifiziert: weight * = $\frac{e}{1-e}$
- Normalisiere Gewichte

Gewichtung: $\frac{e}{1-e}$ → kleine Gewichte für gute Vorhersagen

Klassifizierung:

Gewichte die Modell gemäß ihres Fehlers e :

$$-\log\left(\frac{e}{1-e}\right)$$

→ gute Classifier bekommen hohe Gewichte

→ gewichtete Abstimmung über Label

Stacking

jetzt: erlaube versch. Modelltypen

Meta-Learner kombiniert Vorhersagen

Level 0 - Modelle: zugrunde liegende Modelle

Level 1 - Modell: Metamodell

↪ verwendet Hold-Out-Set um Level-0 Modelle etwas zu verbessern

Theory-Guided Data Science

Scientific Data:

- Ziel: Haupt-Zusammenhänge, Ursachen erkennen
viele komplexe Variablen, Zusammenhänge schwer erkennbar
- oft zu wenige Daten für automatische Modellerstellung

Internet Data:

- deutlich größerer Umfang
- oft gute Resultate durch Verfahren ohne Domänenwissen
- Ziel: meistens Service bereitstellen

Domänenwissen

- Experten haben schon Modelle
- - oft zu simple Modelle
• können auch falsch sein

physically consistent Models

- Modell / Vorhersage ist konsistent mit Naturgesetzen oder bekannten wissenschaftl. Zusammenhängen

Ansätze

Model structure Auswahl

- Domänenwissen kann Hinweis auf geeignete Modellstruktur geben
- Problem modularisieren → Modell für jedes Subproblem

Lineare Regression

$$\mu = \omega^T \cdot x + b$$

(-) Fehler normalverteilt

Punkte nicht unbedingt auf einer Linie / Ebene

Generalized Linear Model

- Abh. Eingabe - Ausgabe nicht unbed. linear
- versch. Fehlerverteilungen möglich

↪ GLM: Link function g

$$g(\mu) = m^{-1}(\mu) = \eta \quad m = \text{linear predictor}$$

Modell verbessern

- Domänenwissen nutzen um Modelle zu verbessern

Initialisierung der Modellparameter

- zB seeds bei k-means "von Hand" wählen
- Verfahren: Matrix Completion
 - bisher: leere Zellen initialisieren \Rightarrow zufällig + iterativ, ... auffüllen

JETZT: Domänenwissen nutzen um manche Zellen schon richtig zu befüllen

Modell einschränken

Bayes Netzwerke: Graph-Repr. von Variablen und deren Beziehungen

- ↪ Domänenwissen: nur Attribute paaren wenn beim Training berücksichtigt werden sollen

Constraints: Constraints mit Domänenwissen spezifizieren

Physical consistency garantieren

- physical consistency nutzen um Modelle einzuschränken / Möglichkeiten auszuschließen

Regularisierung

- einfache Modelle bevorzugt

- Lasso: Qualitätsmaß \rightarrow Güte der Vorhersage + # nötiger Variablen
 - ↪ Variablen beim Training gruppieren (Domänenwissen)

Ergebnis verbessern

Hybride Modelle

- nutze bestehende Modelle
- vergleiche mit trainierten Modellen und tatsächlichem Verhalten