Applied Analytical Data Science

Teil 8: Multi-relationale Verfahren

Dr. Jörg-Uwe Kietz,
Vorlesung an der Univ. Zürich,

Mittwoch, 14:00-15:45 Uhr Vorlesung,
16:00-17:30 Uhr Übung

http://www.kietz.ch/AADS/

Heutiges Programm

- Das Problem der multi-relationalen Daten
- Prädikatenlogik (First Order Logic, FOL),
  - FOL und Datenbanken (Deductive databases)
  - FOL und Data Mining (Inductive logic programming, ILP)
- Beschreibungs-Logiken (Description Logic, DL)
  - DL und Datenbanken (Object-oriented databases)
  - DL und Data Mining
- Aufbereitung der multi-relationalen Daten

Daten in mehreren Relationen

- DM-Ziel z.B Klassifikation der Kunden in ABC-Kunden.
- Alle bisherigen DM-Verfahren arbeiten auf nur einer Tabelle.
- Nur mit der Kundentabelle ist das Ziel nicht zu erreichen.
- Es braucht Informationen aus den anderen Tabellen.
- Wie bekomme ich alle notwendigen Daten in das DM Tool?

Heutiges Programm

- Das Problem der multi-relationalen Daten
- Prädikatenlogik (First Order Logic, FOL),
  - FOL und Datenbanken (Deductive databases)
  - FOL und Data Mining (Inductive logic programming, ILP)
- Beschreibungs-Logiken (Description Logic, DL)
  - DL und Datenbanken (Object-oriented databases)
  - DL und Data Mining
- Aufbereitung der multi-relationalen Daten

Relationen & FOL

Tabellen entsprechen Fakten zu einem Prädikat:

id	shoe	cheese	caravan	tourist
1	unknown	gouda	yes	no
2	unknown	camenbert	yes	yes
3	wood	gouda	yes	no
4	normal	gouda	no	yes
5	wood	camenbert	yes	no
6	unknown	gouda	no	no
7	normal	gouda	yes	yes
8	wood	gouda	no	no
9	unknown	camenbert	no	yes
10	normal	camenbert	yes	yes
11	unknown	gouda	no	???

Deklaration:

person(id,shoe,cheese,caravan,tourist).

Fakten:

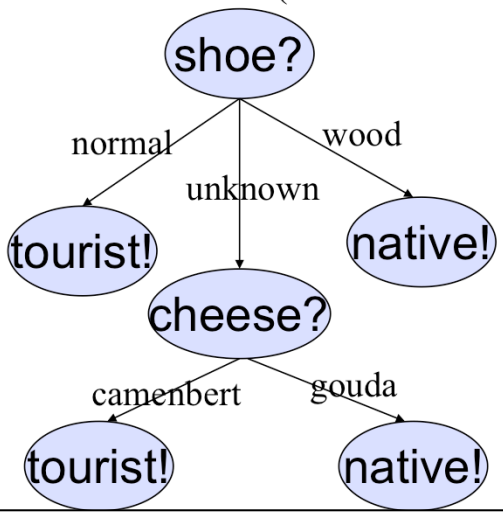
person(1,unknown,gouda,yes,no).
person(2,unknown,camenbert,yes,yes).
person(3,wood,gouda,yes,no).
....

Anfragen:

?- person(ID,_,_,_,yes).
(gibt mir die ID eines Touristen)

Relationen & FOL

Ein Klassifikator (z.B. Entscheidungsbaum) enspricht Regeln




```
person(ID,Shoe,_,_,yes):-
    Shoe = normal.
person(ID,Shoe,Cheese,_,yes):-
    Shoe = unknown,
    Cheese = camenbert.
person(ID,Shoe,Cheese,_,no):-
    Shoe = unknown,
    Cheese = gouda.
person(ID,Shoe,_,_,no):-
    Shoe = wood.
```

Klassikfikation:

```
?- person(11,unknown,gouda,no,Tourist).
    Yes: Tourist = no
```

Prädikatsdefinitionen mit Sorten und Modes

- Sorten sind entweder disjunkt (keine gemeinsamen Terme) oder in einer Sortenhierarchie (Teilmengenbeziehung) angeordnet.
  z.B. Sorten: bpid, bid, aid, number
  z.B. Sortenhierarchie integer < number
  => Eine Variable darf nicht bei disjunkten Sorten auftauchen
- Modes spezifizieren die „Benutzung“ eines Prädikates:
  z.B. „+“: Das Argument muss beim Aufruf instanziert sein.
  „-“: Das Argument wird durch den Aufruf instanziert.
  „#“: Das Argument liefert eine zu lernende Konstante.
- Prädikatsdefinitionen definieren den Typ der Argumente und die Art der Benutzung, z.B. bestellpos(-bpid,+bid,-aid,#number, ...).

Relationen & FOL Beispiel

kunde(+kid,#abc, ...).
bestellung(-bid, +kid, #date, ...).
bestellpos(-bpid,+bid,-aid,#number, ...).
artikel(+aid, #warengrp,#number, ...).

kunde(K,a, ...):- bestellung(B,K,_,...), bestellpos(_,B,A,M, ...),
artikel(A,_,P,...), P * M > 10‘000

Klassifikationslernen in ILP (Idealisiert)

- Gegeben:
  - $B \in LB$ (Hintergrundwissen)
  - $E = E^+ \cup E^- \in LE$ (Beispiele), so dass $B, E \not\models \bot$ und $B \not\models E$.
  - Eine Sprache für Hypothesen LH (Regelmengen)
- Gesucht eine Hypothese $H \in LH$, so dass:
  - H ist konsistent: $B, H, E \not\models \bot$.
  - H leitet mit B die positiven Beispiele in E^+ab: $B, H \models E^+$.
  - H leitet mit B kein negatives Beispiel aus E^- ab: $B, H \not\models E^-$.
- Das System antwortet "Nein“, wenn kein solches H existiert.

Ein ILP-Beispielproblem

Beispiele

minor_violation(event1).
minor_violation(event2).
minor_violation(event3).
minor_violation(event4).

Hintergrundwissen

car_parked(event1,place1).
bus_lane(place1).
car_parked(event2,place2).
second_row(place2).
involved_vehicle(event3,car1).
corrosion(car1).
no_parking(X) ← bus_lane(X).
no_parking(X) ← second_row(X).

Hypothesen

minor_violation(X) ← car_parked(X,Y), no_parking(Y).
minor_violation(X) ← involved_vehicle(X,Y),corrosion(Y).

Vorteile von ILP

+ Das ILP-Framework passt sehr gut zu relationalen Datenbanken (z.B.: Deduktive Datenbanken)
+ Das ILP-Framework ist viel mächtiger als normale Data Mining Verfahren (Prädikatenlogik statt Aussagenlogik).
+ Es gibt ILP-Verfahren für alle Data Mining Aufgaben:
  - Klassifikation (Regeln und Entscheidungsbäume)
  - Regression
  - Assoziationsregeln (in ILP z.B. Datenbank-Constraints)
  - Subgruppenentdeckung
  - Clustering

Nachteile von ILP

- Die höhere Ausdrucksfähigkeit hat ihren Preis:
  - Der Hypothesentest (Beweisbarkeit, Klassifikation) ist viel aufwendiger.
  - Das Finden von konsistenten Hypothesen (Data Mining) ist noch aufwendiger.
  - Die „Kurzsichtigkeit“ von Heuristiken ist extremer, d.h. Heuristiken führen noch viel öfter zu einem lokalen statt einem globalen Optimum.
- ILP erweitert die Form der möglichen Hypothesen nur in einer Richtung: existentielle Anfragen, es gibt z.B. keine Aggregationen

Komplexität des Hypothesentests

- Die logische Folgerung (z.B. Resolution für Klauseln, SLD-Resolution für Regeln) ist im Allgemeinen **unentscheidbar**.
- Für **Datalog** (keine Funktionen) ist sie entscheidbar, aber:
- Selbst für zwei beliebige nicht selbst resolvierende Datalog Klauseln, bzw. nicht rekursive Datalog-Regeln ist die Folgerung (**θ-Subsumption**) **NP-Vollständig** (d.h. alle bekannten Verfahren haben einen exponentiellen Zeitaufwand.)
- **θ-Subsumption** ist polynomial für **deterministische Regeln**
- **θ-Subsumption** ist polynomial für ***k*-lokale Regeln**

θ-Subsumption

Eine Klausel D θ-subsumiert eine Klausel C für eine Substitution θ

$$\mathbf{D} \models_{\theta} \mathbf{C} :\Leftrightarrow \exists\theta: \mathbf{D}\theta \subseteq \mathbf{C} \text{ (Robinson,65)}$$

Beispiel:

D = +(S) ← part(S,O1), big(O1), part(S,O2), black(O2).

C = +(s1) ← part(s1,rect1), big(rect1), black(rect1).

D $\models_{\theta}$ C mit θ = { S/s1, O1/rect1, O2/rect1}

θ-Subsumption ist eine korrekte Ableitungsrelation:

D $\models_{\theta}$ C ⇒ D $\models$ C

θ-Subsumption

θ-Subsumption ist unvollständig für rekursive Regel:

- D = path(X,Z) ← link(X,Y), path(Y,Z)
- C = path(X,Z) ← link(X,Y1), link(Y1,Y2), path(Y2,Z)
- D |= C, aber D $|\!\!-\!\!/\!\!-_\theta$ C

θ-Subsumption ist eine vollständige Ableitungsrelation, gdw.
D nicht selbstresolvierend (rekursiv) und
C keine Tautologie ist.

Zur Benutzung der θ-Subsumption müssen die Beispiele mit dem Hintergrundwissen vervollständigt (saturiert) werden.
(Wenn B generativ ist, dann) B,H |= E, gdw. H $|\!\!-_\theta$ saturated$_B$(E)

Vervollständigung von Beispielen

Beispiele

minor_violation(event1).
minor_violation(event2).
minor_violation(event3).
minor_violation(event4).

Hintergrundwissen

car_parked(event1,place1).
bus_lane(place1).
car_parked(event2,place2).
second_row(place2).
involved_vehicle(event3,car1).
corrosion(car1).
no_parking(X) ← bus_lane(X).
no_parking(X) ← second_row(X).

Saturierte Beispiele

minor_violation(event1) ← car_parked(event1,place1), bus_lane(place1), no_parking(place1).
minor_violation(event2) ← car_parked(event2,place2), second_row(place2), no_parking(place2).
...

Deterministische Regeln

D ist deterministisch bezüglich C

D = grand_mother(X,Z) ← mother(X,Y),
mother(Y,Z).

C = grand_mother(sue,ann) ← mother(sue,eve),
mother(sue,tom),
mother(eve,ann).

D ist **in**deterministisch bezüglich C

D = grand_mother(X,Z) ← mother(X,Y),
parent(Y,Z).

C = grand_mother(sue,ann) ← mother(sue,eve),
mother(sue,tom),
parent(eve,ann),
parent(frank,ann).

Eine Regel ist deterministisch, gdw. sie bezgl. allen Beispielen deterministisch ist.

Deterministische Regeln

D = grand_mother(X,Z) ← mother(X,Y),
mother(Y,Z).

C = grand_mother(sue,ann) ← mother(sue,eve),
mother(sue,tom),
mother(eve,ann).

- NP-Vollständigkeit von D $|-_\theta$ C kommt aus dem Indeterminismus der Bestimmung von θ.
- Für deterministische Regeln gibt es eine Ordnung der Literale, so dass die Abbildung der Literale die Substitution θ eindeutig bestimmt.

⇒ Backtracking (Ausprobieren und Zurücknehmen) ist nicht notwendig.

⇒ D $|-_\theta$ C kann durch O(|D|*|C|) Unifikationsversuche bestimmt werden.

⇒ Die Datenbank-Entsprechung von deterministischen Regeln sind Joins von 0:1 Relationen.

k-lokale Regeln

Eine Regel ist *k*-lokal, gdw. ihr Körper so aufgeteilt werden kann dass:

- jeder Teil höchstens
  - *k* Literale (*k*-literal lokal), oder
  - *k* freie (nicht im Kopf vorkommende) Variablen (*k*-variable lokal),

  enthält, und
- kein Teil eine freie Variable enthält, die auch in einem anderen Teil vorkommt.

high_risk_driver(D) ← single(D), male(D),
age(D,A), less_than(A,25),
offence(D,O), moving_violation(O),
date(O,Y1), today(Y2), recent(Y1,Y2).

Diese Regel ist,
3-variable-lokal (O,Y1,Y2)
5-literal-lokal (offence-recent)
d.h. 3-lokal (Minimum)

Komplexität des Klassifikationslernens in ILP

Polynomialer Aufwand (+) vs. NP-Schwierigkeit (-) (nicht rekursiv)

+ Klauseln mit maximal *k* Literalen
+ tiefenbeschränkte deterministische Regeln.
- nicht tiefenbeschränkte deterministische Regeln
- tiefenbeschränkte **in**deterministische Regeln, (inklusive 1-variable lokale Regeln)
+ k-literal lokale Horn Regeln

=>

+ tiefenbeschränkte deterministische k-literal lokale Regeln eignen sich zum Data Mining von existentiellen Anfragen an relationale Datenbanken.

Heutiges Programm

- Das Problem der multi-relationalen Daten
- Prädikatenlogik (First Order Logic, FOL),
  - FOL und Datenbanken (Deductive databases)
  - FOL und Data Mining (Inductive logic programming, ILP)
- Beschreibungs-Logiken (Description Logic, DL)
  - DL und Datenbanken (Object-oriented databases)
  - DL und Data Mining
- Aufbereitung der multi-relationalen Daten

Relationen & Objekte

Tabellen entsprechen Instanzen zu einem Concept (Klasse):

id	shoe	cheese	caravan	tourist
1	unknown	gouda	yes	no
2	unknown	camenbert	yes	yes
3	wood	gouda	yes	no
4	normal	gouda	no	yes
5	wood	camenbert	yes	no
6	unknown	gouda	no	no
7	normal	gouda	yes	yes
8	wood	gouda	no	no
9	unknown	camenbert	no	yes
10	normal	camenbert	yes	yes
11	unknown	gouda	no	???

concept: Person(
shoe [1..1]: String {unknown, wood, normal},
cheese [1..1]: String {camenbert, gouda},
caravan [1..1] : String {yes,no}
tourist [1..1] : String {yes,no})

Instanzen: person(1),
1.shoe = unknown
1.cheese = gouda
1.caravan = yes
1.tourist = no.

Anfragen:
?- person(ID) and ID.tourist = yes
(gibt mir die ID eines Touristen)

Relationen & Objekte Beispiel

concept: Kunde(abc[1..1]: String, bestellungen [0..∞]: Bestellung,...).

concept: Bestellung(kunde [1..1]: Kunde, datum [1..1]: Date,
positionen [1..∞]: Bestellpos, ...).

a-kunde := Kunde and
at-least-c(100,bestellungen, BestellungDiesesJahr).

BestellungenDiesesJahr := Bestellung and all(Datum,DiesesJahr).

Beschreibungs-Logiken (Description Logic)

- Beschreibungslogiken haben zwei Sorten von Termen: Concept-terme und Role-Terme.
- Concept-terme ~ 1-stelligen Prädikaten, die wichtigsten sind
  - Concept-Name
  - Concept **and** Concept
  - **all**(Role,Concept)
  - **at-least**(Role)
  - **at-most**(Role)
- Role-Terme ~ 2-stelligen Prädikaten, die wichtigste ist
  - Role-Name

Übersicht über DL-Terme

TERM	INTERPRETATION
TOP-CONCEPT	$\Delta^{\mathcal{I}}$
NOTHING	$\emptyset$
and[C,D]	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
or[C,D]	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
not[C]	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
all[p,C]	$\{d \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
some[p,C]	$\{d \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
at-least[n,p]	$\{d \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(d)\rvert \geq n\}$
at-most[n,p]	$\{d \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(d)\rvert \leq n\}$
at-least-c[n,p,C]	$\{d \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(d) \cap C^{\mathcal{I}}\rvert \geq n\}$
at-most-c[n,p,C]	$\{d \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(d) \cap C^{\mathcal{I}}\rvert \leq n\}$
same-as[p,q]	$\{d \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(d) = q^{\mathcal{I}}(d)\}$
subset[p,q]	$\{d \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(d) \subseteq q^{\mathcal{I}}(d)\}$
not-same-as[p,q]	$\{d \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(d) \neq q^{\mathcal{I}}(d)\}$
fills[p,b]	$\{d \in dom^{\mathcal{I}} \mid b^{\mathcal{I}} \in p^{\mathcal{I}}(d)\}$
not-fills[p,b]	$\{d \in dom^{\mathcal{I}} \mid b^{\mathcal{I}} \notin p^{\mathcal{I}}(d)\}$
one-of[b_1,...,b_m]	$\{b_1^{\mathcal{I}}, \ldots, b_m^{\mathcal{I}}\}$

TERM	INTERPRETATION
TOP-ROLE	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
IDENTITY	$\{(d,d) \mid d \in \Delta^{\mathcal{I}}\}$
role-and[p,q]	$p^{\mathcal{I}} \cap q^{\mathcal{I}}$
role-or[p,q]	$p^{\mathcal{I}} \cup q^{\mathcal{I}}$
role-not[p]	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$
inverse[p]	$\{(d,d') \mid (d',d) \in R^{\mathcal{I}}\}$
restrict[p,C]	$\{(d,d') \in p^{\mathcal{I}} \mid d' \in C^{\mathcal{I}}\}$
compose[p,q]	$p^{\mathcal{I}} \circ q^{\mathcal{I}}$
product[C,D]	$C^{\mathcal{I}} \times D^{\mathcal{I}}$
trans[p]	$\bigcup_{n>1}(p^{\mathcal{I}})^n$

Beschreibungs-Logiken

- DL-Terme können zu komplexen Ausdrücken zusammengesetzt werden, z.B.:
  - Kunde and at-least(100, bestellungen)
    (Die Menge der Kunden mit mindestens 100 Bestellungen)
  - Kunde and at-least-c(100,bestellungen, all(Datum,DiesesJahr)).
- Concept-terme können durch „:<“ und „:=“ Axiome benannt werden:
  - Kunde :< all(bestellungen,Bestellung) and at-least(1,abc)
    all(abc, one-of({a,b,c}) and and at-most(1,abc).
  - a-kunde := Kunde and at-least-c(100,bestellungen,
    all(Datum,DiesesJahr)).

Anfragen an DL-Systeme

- Subsumption: C < D, ist C spezieller als D? (C beschreibt notwendigerweise eine Teilmenge der Instanzen von D)
- Konsistenz: Ist C konsistent? (C beschreibt nicht notwendigerweise die leere Menge)
- Disjunkt: Sind C und D disjunkt? (Es kann keine gemeinsamen Instanzen von C und D geben)
- Äquivalenz: C = D, sind C und D gleich? (Haben C und D notwendigerweise dieselben Instanzen)

=> Alle Anfragen lassen sich auf Subsumption zurückführen.

- Realisation: i ∈ C, ist i eine Instanz von C.
- Abstraktion: Das speziellste C zu i (i∈C, ¬∃D, mit i∈D und D<C)

Komplexität von Beschreibungs-Logiken

- In Beschreibungslogiken ist die Balance (Trade-Off) zwischen Ausdrucksstärke und Komplexität des Beweisens (Subsumption zwischen Termen) sehr gut untersucht, je nach verwendeten Termen reicht die Komplexität von unentscheidbar bis polynomial:

CONCEPT CONSTRUCTORS	*ROLE CONSTRUCTORS*	*COMPLEXITY*
and,all, same-as	—	Undecidable [59]
$\mathcal{ALCNR}$ (and, or, not, some, at-least, at-most)	role-and	PSPACE [35], [34]
and, not, all, some, or	compose, role-or, inverse, trans, restrict,TOP-ROLE	EXP-TIME [58]
and, all, at-most, at-least, same-as on attributes, fills, one-of with integers		Polynomial [19]

Benutzung von DLs

- DLs sind eine **Formalisierung** von „Semantischen Netzen“, „Frames“ und späteren „Objekt-orientierten Formalismen“
- DLs werden benutzt um Ontologien/Terminologien zu modellieren
  - Wissensmanagement => Mining ~ Wissenserwerb
    - WEB-Annotierung => WEB-Mining
    - Dokumentbeschreibungen => Text-Mining
    - Ontologie-Erwerb
- DLs werden benutzt um Objekt-orientierte Datenbanken zu modellieren => Data Mining, Schema-Reengineering

DLs und Data Mining

- Dieses ist ein aktives Forschungsgebiet!
- Einzelne Ansätze DLs zum Klassifikationslernen zu Benutzen gibt es seit 1989, andere Data Mining Tasks (Clustering, Assoziationen, ...) fehlen noch völlig.
- Die Beschreibungs-Logik core-Classic (all, atleast, atmost, eingeschränktes same-as) ist polynomial lernbar.
- Die Beschreibungslogik *ALN* (all, atleast, atmost) ist sehr geeignet, um DB-Schema zu beschreiben und sie stellt eine sehr interessante Ergänzung zu ILP dar.
- Es gibt hybride Ansätze: Lernen von ILP-Regeln mit DL-Termen als Literalen (CARIN-*ALN)*.

Heutiges Programm

- Das Problem der multi-relationalen Daten
- Prädikatenlogik (First Order Logic, FOL),
  - FOL und Datenbanken (Deductive databases)
  - FOL und Data Mining (Inductive logic programming, ILP)
- Beschreibungs-Logiken (Description Logic, DL)
  - DL und Datenbanken (Object-oriented databases)
  - DL und Data Mining

☞ Aufbereitung der multi-relationalen Daten

Aufbereitung der multi-relationalen Daten

- Anstatt multi-relationale Data Mining Verfahren zu benutzen, können die Daten aufbereitet werden, so dass alles **Wesentliche** in einer Tabelle ist.
  - Joins von 0:1 (**deterministischen**) Relationen
  - Spezialisierung von 0:N (**indeterministischen**) Relationen
  - Aggregation von Attributen aus (**indeterministischen**) Relationen.
  - Kombination dieser manuellen Möglichkeiten mit multi-relationalen Data Mining Verfahren!
- Erste Ansätze dieses automatisch zu tun.

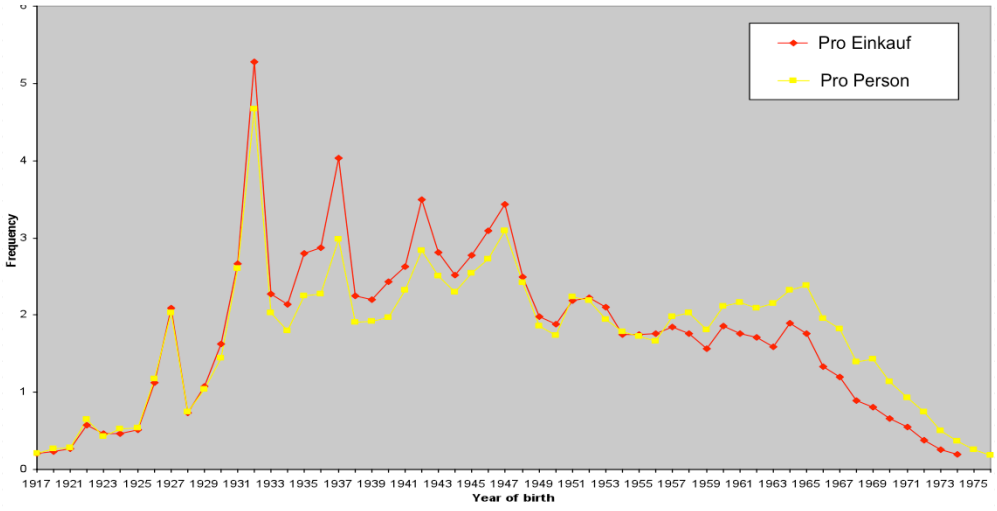
Behandlung Multi-Relationaler Daten (I)

- Joinen von 1:N verknüpftenTabellen, z.B.
  - Personen und ihre Einkäufe und Warenkörbe

\+ Sehr einfach (Universelle Relation)

– Verändert die (Verteilung der) analysierte(n) Population

Joins Verändern die Verteilung



Behandlung Multi-Relationaler Daten (I)

- Joinen von 1:N verknüpftenTabellen, z.B.
  - Personen und ihre Einkäufe und Warenkörbe

+ Sehr einfach (Universelle Relation)
– Verändert die (Verteilung der) analysierte(n) Population
– Keine Gesamtattribute, z.B. Einkäufe pro Woche
– Explodierende Anzahl von Datensätzen
– Objektidentität und structurelle Informationen gehen verloren

Behandlung Multi-Relationaler Daten (II)

- Aggregation von Attributen von 1:N verknüpften Tabellen, z.B.:
  - Anzahl der Einkäufe pro Person,
  - Summe des Ausgegebenen Geldes,
  - Alle Einkäufe finden im selben Laden statt, ...

\+ Eindeutiger join basierend auf der Aggregation

\+ Keine Veränderung der analysierten Population

– Höherer Berechnungsaufwand (SQL: group by)

– Gefahr der Aggregation von“Äpfeln" und “Birnen"

– Keine individuellen Attribute

Behandlung Multi-Relationaler Daten (III)

- Spezialisierung der 1:N Relation in deterministische (1:1) Relationen, e.g.:
  - Einkauf an einem bestimmten Tag
  - Der letzte (vorletzte, ...) Einkauf

+ Eindeutige Joins bei 1:1 Spezialisierungen
+ Keine Veränderung der analysierten Population
– Höherer Berechnungsaufwand (mehrere Joins)
– Nicht immer einfach bedeutsame Spezialisierungen zu finden
– Explodierende Anzahl von Attributen mit vielen unbekannten Werten

Behandlung Multi-Relationaler Daten (final)

Kombination von (II) und (III) und Segmentierung:

1) Spezialisiere 1:N Relationen sofern Bedeutungstragend
2) Bilde/finde strukturierte Prototypen aus/in den Daten
3) Segmentiere die Daten gemäss dieser strukturierten Prototypen
4) Joine die deterministischen (1:1) Relationen
5) Aggregiere die immer noch indeterministischen Relationen

Mehr in der Vorlesung über Multi-relationales Data Mining

Fazit

- Daten liegen oft in mehreren verknüpften Tabellen vor, aber “Normale” Data Mining Verfahren akzeptieren nur eine Tabelle.
- Insb. bei MRDM bestimmt die Data Mining Aufgabe, welche Daten wie aufbereitet ins Data Mining einfliessen sollen.
- Mit ILP (-> deduktive DBs) und DL (-> OO-DBs) gibt es auch Verfahren zum Multi-Relational Data Mining (MRDM).
- Es gibt einen Trade-Off zwischen Ausdrucksfähigkeit und Komplexität der Data Mining Verfahren.
- Ein „dekarativer Bias“ (Sorten, Modes, Klausel-Aufbau, ...) soll helfen das Machbare auf das Interessante zu fokussieren.

Literatur

Literatur zu Inductive Logic Programming:

- Dzeroski, S.: „Inductive Logic Programming and Knowledge Discovery“, In: Fayyad, U.; Piatetetsky-Shapiro, G.; Smyth, P.; Uthurusamy, R.: Advances in Knowledge Discovery and Data Mining, AAAI Press / The MIT Press, 1996.
- Kietz, J.-U.: „Induktive Analyse Relationaler Daten“, Doktorarbeit, TU-Berlin, 1997.

Literatur

Literatur zu Beschreibungs-Logiken:

- Borgida, A.: Description Logics in Data Management“, IEEE Transactions on Knowledge and Data Engineering, Vol 7, 1995.
- Cohen, W.; Hirsh, H.: „The Learnability of Description Logics with Equality Constraints“, Machine Learning Journal, Vol 17, 1994.
- Kietz, J.-U, Morik, K.: „A polynomial approach to the constructive Induction of Structural Knowledge“, Machine Learning Journal, Vol 14, 1994.
- Rouveirol, C.; Ventos, V.: „Towards Learning in CARIN-ALN“, Proc. of the ILP-2000, 2000.