Aspekte des Maschinellen Lernens

- **Bisher sind wir davon ausgegangen, dass Wissensbasen von einem „Knowledge Engineer“ erstellt werden**
- **Höchst aufwändiger und kostenintensiver Prozess**
- **Warum nicht einfach Beispiele vorgeben und Wissensbasis vom System selbst generieren, also lernen, lassen?**
- **Maschinelles Lernen aktives und wichtiges Teilgebiet der KI**
- **Hier nur einige Aspekte herausgegriffen:**

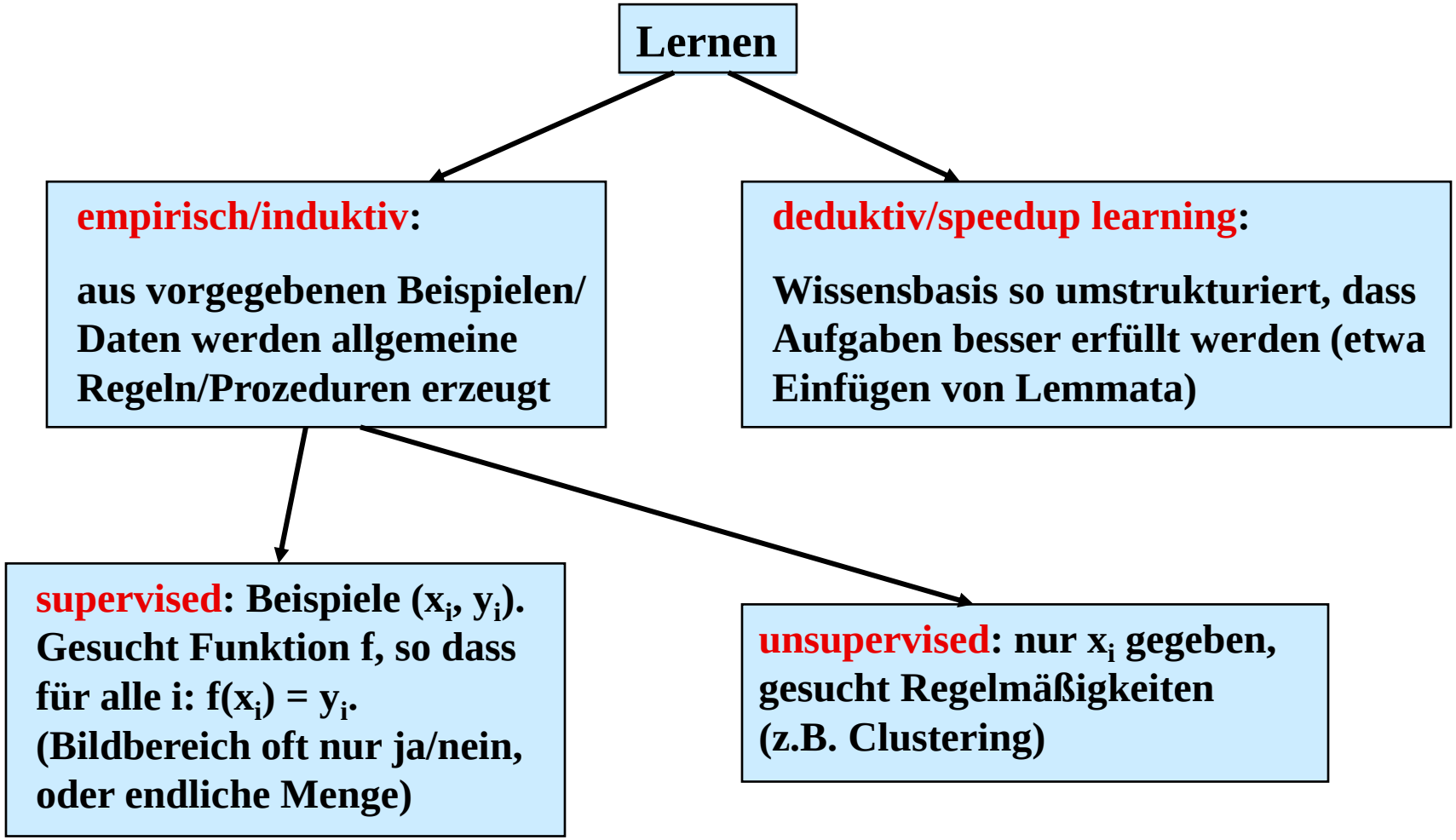
1. **Lernen von Entscheidungsbäumen**
2. **Versionsräume**
3. **Induktives Logisches Programmieren**
4. **Instanzbasiertes Lernen**

Was ist Lernen?

Einige Definitionen von bedeutenden Leuten:

- **Simon: jede Veränderung eines Systems, die es ihm erlaubt, eine Aufgabe bei Wiederholung derselben Aufgabe oder einer Aufgabe derselben Art besser zu lösen.**
- **Scott: ein Prozess, bei dem ein System eine abrufbare Repräsentation von vergangenen Interaktionen mit seiner Umwelt aufbaut.**
- **Michalski: das Konstruieren oder Verändern von Repräsentationen von Erfahrungen.**

Klassifikation des Lernens



Lernen aus logischer Sicht

gegeben: **Hintergrundwissen B**

Menge möglicher Hypothesen Hyp

Beschreibungen der Beispiele D

Klassifikation der Beispiele C, dabei häufig Unterscheidung in positive (C+) und negative Beispiele (C-)

gesucht: **Hypothese H aus Hyp, so dass $B \wedge H \wedge D \wedge C-$ konsistent und**

$$B \wedge H \wedge D \models C+$$

Beispiel: **B: Amsel(x) $\supset$ Vogel(x), Spatz(x) $\supset$ Vogel(x)**

Hyp: beliebige Konjunktionen von Hornklausen

D: Amsel(O_1), Spatz(O_2), Hund(O_3), Katze(O_4), Hamster(O_5)

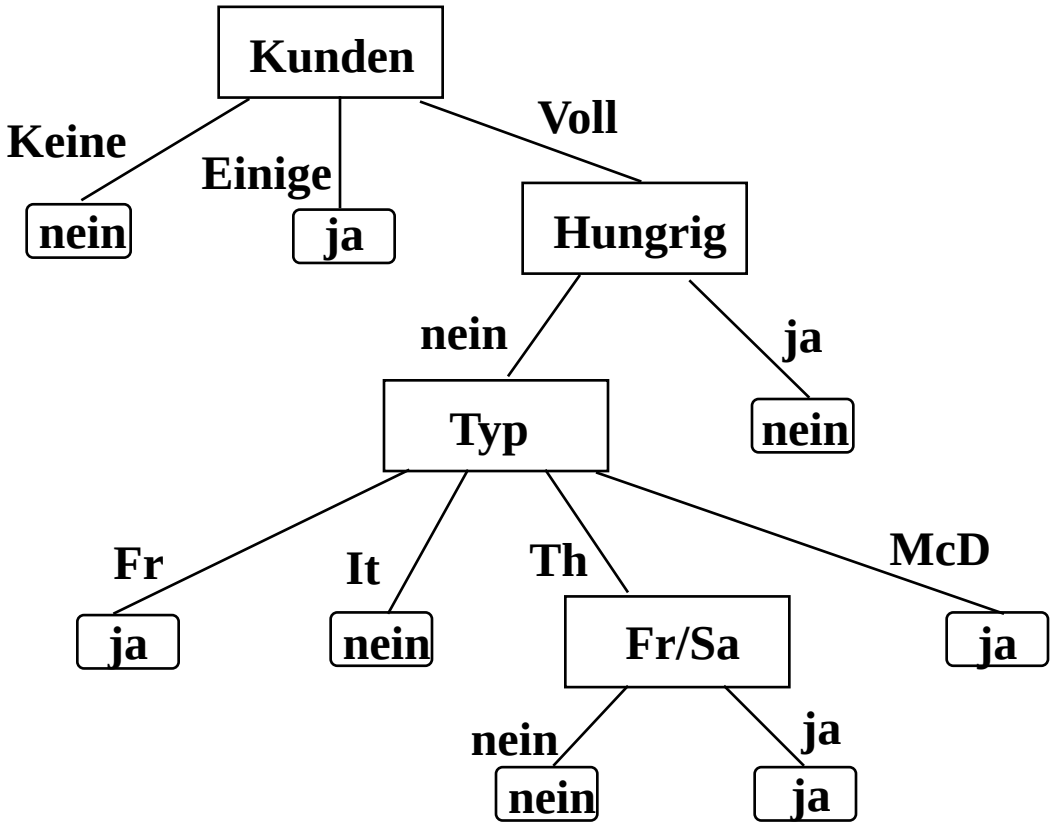
C+: Fliegt(O_1), Fliegt(O_2) C-: ¬Fliegt(O_3), ¬Fliegt(O_4), ¬Fliegt(O_5)

H: Vogel(x) $\supset$ Fliegt(x)

Entscheidungsbäume

gegeben: Menge von Eigenschaften, die Situation beschreiben
Entscheidungsbaum zeigt, ob Zielprädikat erfüllt ist oder nicht

Im Restaurant auf Platz warten?



Entscheidungsbäume, ctd.

logisch gesehen sind Entscheidungsbäume Mengen von Implikationen

Kunden = Voll ∧ ¬ Hungrig ∧ Typ = It ⊃ ¬Warten,

Kunden = Einige ⊃ Warten, etc.

gegeben: Menge von positiven und negativen Beispielen

gesucht: möglichst einfacher mit Beispielen konsistenter Entscheidungsbaum

Beispiel	Kunden	Hungrig	Typ	Fri/Sat	Ziel
1	Einige	N	Fr	N	J
2	Voll	N	Th	N	N
3	Einige	J	McD	N	J
4	Voll	N	Th	J	J
5	Voll	J	Fr	J	N
6	Einige	N	It	N	J
7	Keine	J	McD	N	N
8	Einige	N	Th	N	J
9	Voll	J	McD	J	N
10	Voll	N	It	J	N
11	Keine	J	Th	N	N
12	Voll	N	McD	J	J

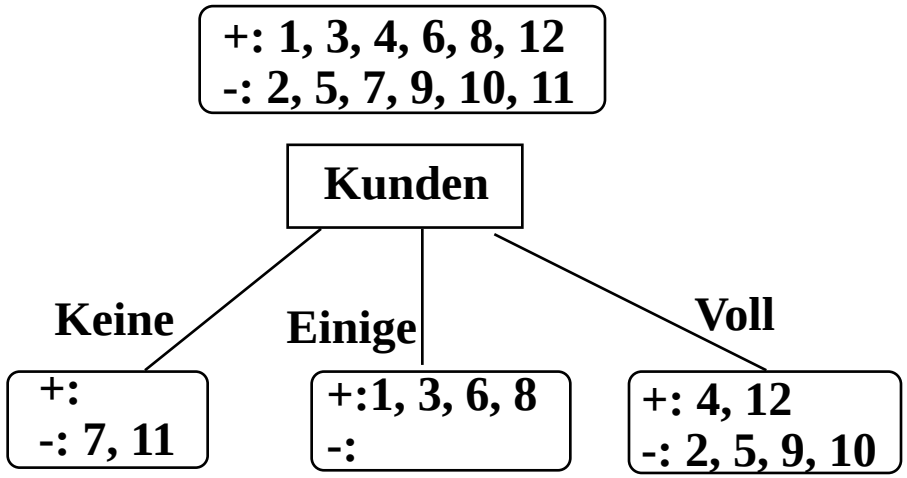
Generieren des Entscheidungsbaums

- **Rekursiver Algorithmus**
- **verwendet als Test Attribut, dessen Werte die Beispiele "am besten" in nur positive, nur negative, und gemischte Mengen aufspaltet**
- **gibt es nur noch positive => Antwort Ja**
- **gibt es nur noch negative => Antwort Nein**
- **gibt es noch gemischte Mengen => rekursiver Aufruf des Alg. mit restlichen Beispielen und verbleibenden Attributen**
- **gibt es an einem Knoten keine Beispiele mehr => verwende Default-Antwort**
- **gibt es noch undifferenziertes Beispiel, aber keine Attribute mehr => Problem: entweder nicht genug Attribute, oder Fehler in den Daten**

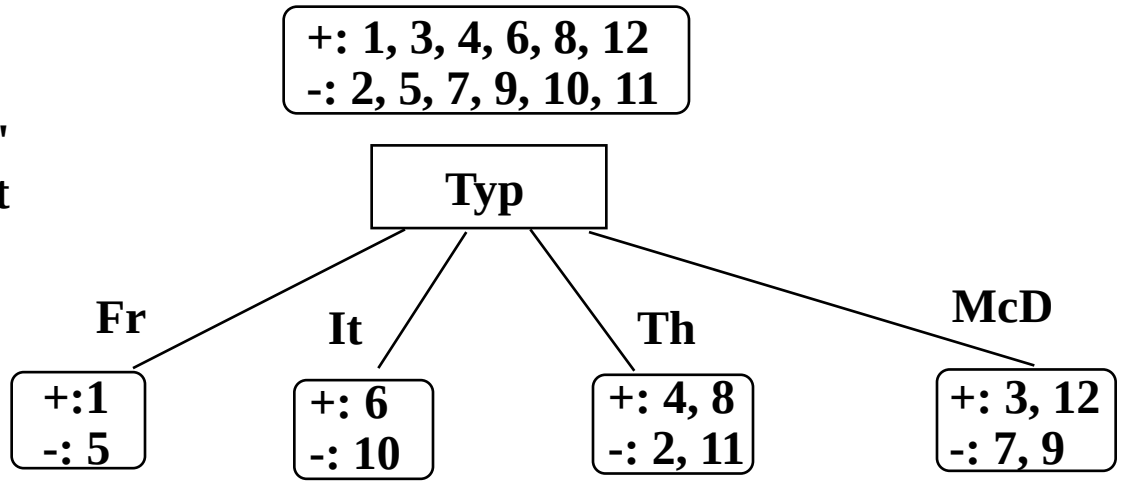
welches Attribut "am besten" ist, sagt z.B. die Informationstheorie (hier intuitives Verständnis ausreichend)

Unser Beispiel

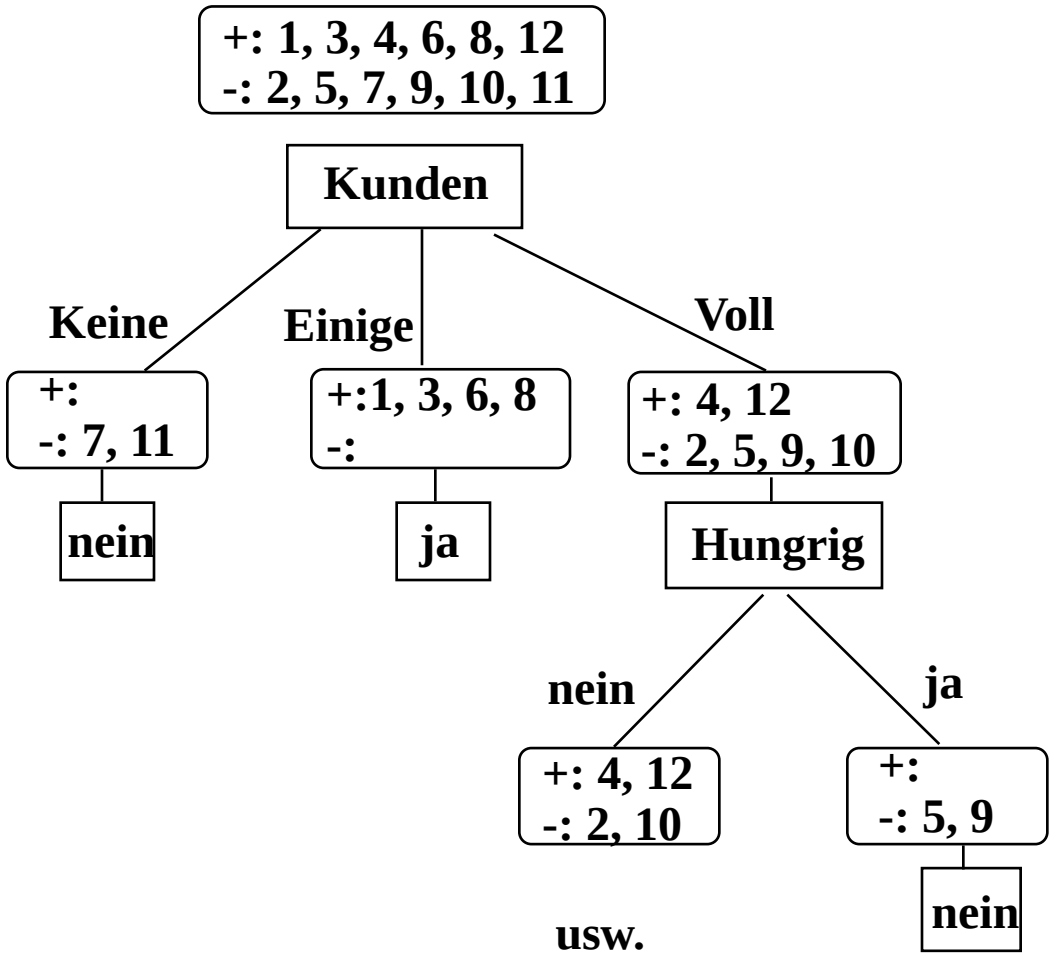
ein "gutes" Anfangsattribut



ein "schlechtes" Anfangsattribut



Beispiel, ctd



Lernen mit Versionsräumen (version space learning)

- **Beim Lernen von Entscheidungsbäumen können neue Beispiele vollständige Rekonstruktion des Baumes zur Folge haben**
- **Grund: es wird immer genau eine hypothetische Definition des Zielprädikats konstruiert**
- **beim VSL werden ALLE mit den bisherigen Beispielen konsistenten Hypothesen generiert**
- **unter Hypothese verstehen wir hier eine mögliche Definition des Zielprädikats in der dafür zugelassenen Sprache**

Beispiel:

$$\forall\, r\ \text{Warten}(r) \Leftrightarrow$$
$$\text{Kunden}(r, \text{Einige}) \vee$$
$$\text{Kunden}(r, \text{Voll}) \wedge \neg\text{Hungrig}(r) \wedge \text{Typ}(r, \text{Fr}) \vee$$
$$\text{Kunden}(r, \text{Voll}) \wedge \neg\text{Hungrig}(r) \wedge \text{Typ}(r, \text{Th}) \wedge \text{Fr/Sa}(r) \vee$$
$$\text{Kunden}(r, \text{Voll}) \wedge \neg\text{Hungrig}(r) \wedge \text{Typ}(r, \text{McD})$$

Beispiele sind Formeln:

$$\text{Kunden}(X_1, \text{Einige}),\ \text{Hungrig}(X_1),\ \text{Typ}(X_1, \text{Fr}),\ \neg\text{Fr/Sa}(X_1),\ \text{Warten}(X_1)$$

VSL, ctd.

- **VSL startet mit der Menge aller Hypothesen H1, ..., Hn und benutzt Beispiele, um nach und nach alle inkonsistenten Hypothesen zu eliminieren**
- **die Beschreibung der konsistenten Hypothesen verwendet die allgemeinste Grenzmenge (most general boundary set, G-set) und die spezifischste Grenzmenge (most specific boundary set, S-set)**

  $\forall x$ P(x) <=> C1(x) ist (echt) allgemeiner als $\forall x$ P(x) <=> C2(x) genau dann wenn $\forall x$ C2(x) => C1(x) und nicht umgekehrt

  $\forall x$ P(x) <=> C1(x) ist (echt) spezifischer als $\forall x$ P(x) <=> C2(x) genau dann wenn $\forall x$ C1(x) => C2(x) und nicht umgekehrt

Bsp.: $\forall x$ Schön(x) <=> Porsche(x) allgemeiner als

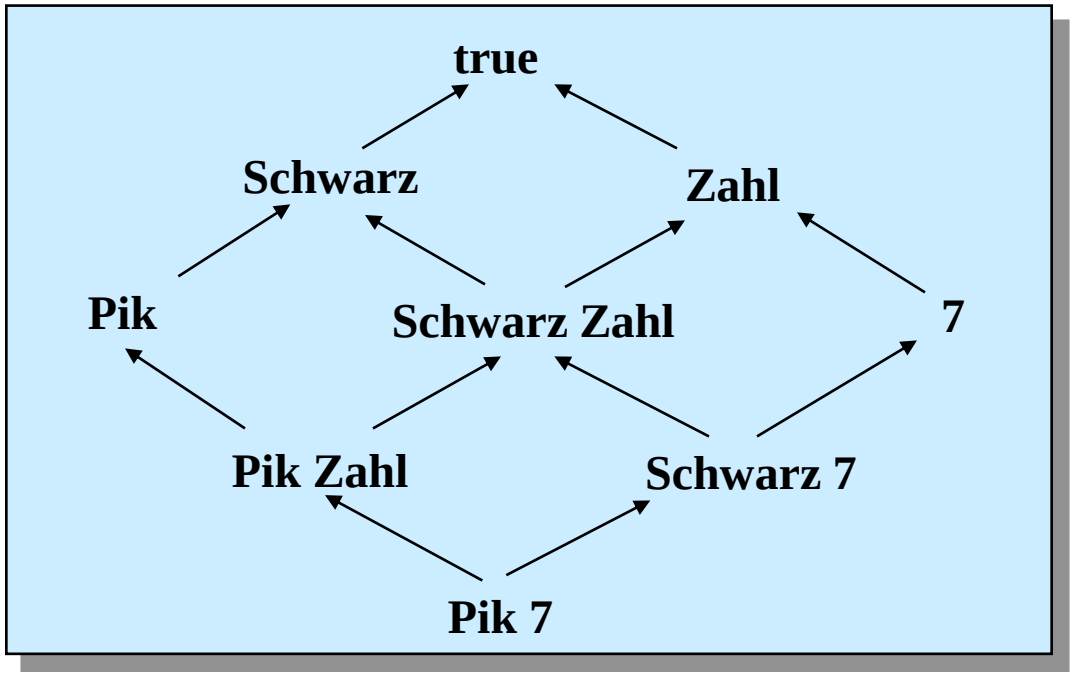
$\forall x$ Schön(x) <=> Porsche(x) $\wedge$ Rot(x)

- **jedes Element H des S-set ist konsistente Hypothese, und es gibt keine Hypothese, die konsistent und spezifischer als H ist**
- **jedes Element H des G-set ist konsistente Hypothese, und es gibt keine Hypothese, die konsistent und allgemeiner als H ist**

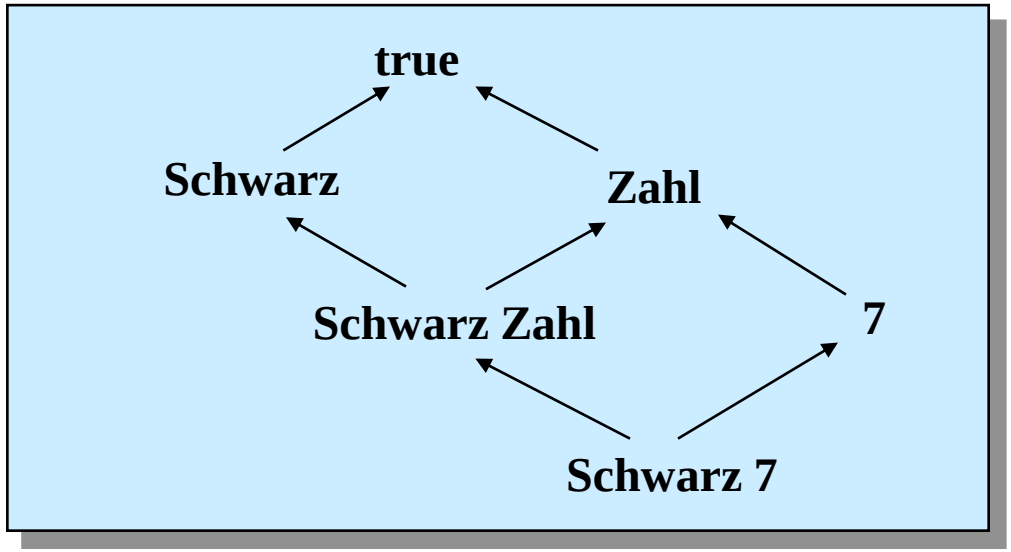
Ein Kartenbeispiel

- **Sprache für Hypothesen: true, false + konsistente Konjunktionen aus:**

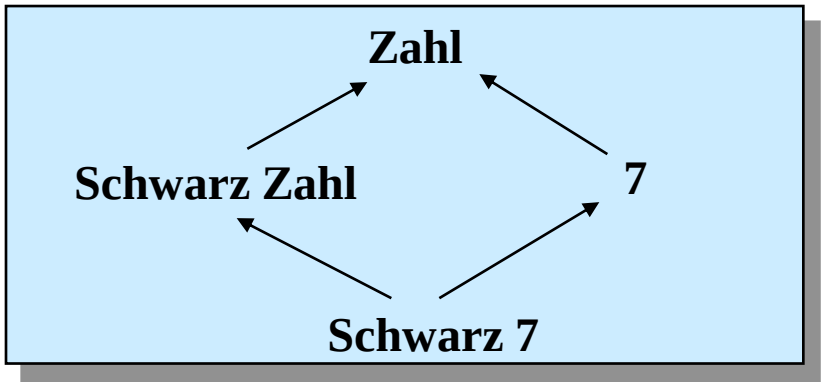
  7, 8, 9, 10, B, D, K, A, Zahl, Bild, Pik, Kreuz, Karo, Herz, Rot, Schwarz
- **übliche Spezifizitätsbeziehungen sollen gelten (7 ist Zahl, Pik ist Schwarz etc.)**
- **Versionsraum nach Eingabe von Pik 7 als positives Beispiel, Pfeile gehen zu nächst allgemeinerer Hypothese:**



Versionsraum nach Eingabe von Kreuz 7 als pos. Instanz:



und nach Eingabe von Pik Dame als neg. Instanz:



Versionsraum-Algorithmus

Starte mit G-set {true} und S-set {false}.

Eingabe eines neuen Beispiels => führe (evtl. mehrfach) folgende Regeln aus:

- **falls neues negatives Beispiel von Element in S-set fälschlich positiv bewertet wird, so eliminiere dieses Element aus dem S-set**
- **falls neues positives Beispiel von Element in S-set fälschlich negativ bewertet wird, so ersetze dieses Element durch alle seine direkten Generalisierungen**
- **falls neues positives Beispiel von Element in G-set fälschlich negativ bewertet wird, so eliminiere dieses Element aus dem G-set**
- **falls neues negatives Beispiel von Element in G-set fälschlich positiv bewertet wird, so ersetze dieses Element durch alle seine direkten Spezialisierungen**

Problem: Fehler in den Daten führt zu Kollaps

- **Hypothesenraum muss syntaktisch beschränkt werden, sonst enthält S-set nur die Disjunktion aller positiven Beispiele und G-set die Negation der Disjunktion der negativen Beispiele**

Induktive Logikprogrammierung

- **gegeben:**
  Menge positiver Beispiele P für Ziel g(X1, ..., Xk),
  Menge negativer Beispiele N für Ziel g(X1, ..., Xk).
- **sind jeweils Fakten der Gestalt g(c1, ..., ck)**
- **Hintergrundwissen B (definites Logikprogramm)**
- **gesucht: Menge H von Regeln, die Zielprädikat definieren:**
  B ∪ H |- p für alle p aus P; B ∪ H |- n für kein n aus N

Algorithmen

```
find_hypothesis(P,N)
H := {};
while P ≠ {} do
        r := find_a_rule(P,N);
        H := H ∪ {r};
        P := {p ∈ P | nicht B ∪ H |- p }
output H.
```

liefert Regelmenge H für pos. Beispiele P und negative N

Algorithmen, ctd.

```
find_a_rule(P,N)
head := g(X1,...,Xk);          % g Zielprädikat, k seine Stelligkeit
body := true;
while B ∪ {head ← body}|- n für ein negatives Beispiel n do
        l := zulässiges Literal lit, das H(head ← body, lit) maximiert;
        body := body, l;
output(head ← body)
```

- Alg. liefert Regel, die einige positive Beispiele herleitet, kein negatives.
- H ist der heuristische Wert einer Regel, etwa:
  # positive erfasste Beispiele - # negative erfasste Beispiele
- Suchraum kann immens groß werden, deshalb: Einschränkung der für Regelkörper zulässigen Literale essentiell.

Beispiel

Hintergrundwissen B:

parent(ann,mary). parent(ann,tom).
parent(tom,eve). parent(eve,ian).
female(ann). female(mary). female(eve)

Beispiele:

P: daughter(mary,ann). daughter(eve,tom).
N: daughter(tom,ann). daughter(eve,ann).

Annahmen:

heuristischer Wert: Anzahl positiver abgedeckter – Anzahl negativer

zulässige Literale: solche, in denen nur Variablen als Argumente vorkommen

Aufruf von find_a_rule(P,N), konstruierte bodies

body1: initialisiert mit true, alle negativen Beispiele herleitbar aus B und

daughter(X1,X2) ← true

body2:

Kand.:	female(X1),	female(X2),	parent(X1,X2),	parent(X2,X1)
H:	2-1	1-2	0-0	2-1

Alternative 1 und 4 gleich gut, nehmen wir female(X1)

daughter(eve,ann) ∈ N aus B und daughter(X1,X2) ← female(X1) abgeleitet

body3:

Kand.:	female(X2),	parent(X1,X2),	parent(X2,X1)
H:	1-1	0-0	2-0

max. heuristischen Wert für parent(X2,X1), kein neg. B. abgeleitet aus B und

daughter(X1,X2) ← female(X1), parent(X2,X1)

diese Regel wird ausgegeben; da sie alle positiven Beispiele abdeckt -> fertig

Instanzbasiertes Lernen

Bisher: Lernen der gesuchten Funktion/Klassifikation durch Erzeugen allgemeiner Regeln:

Beispiele:	➔	Regeln:
(x_1, y_1)		$y_1(x)$:- Bedingung
...		...
(x_n, y_n)		$y_m(x)$:- Bedingung

Wenn neues Objekt x_i auftaucht: y-Wert durch erzeugte Regeln ermittelt.

Kann man nicht den Wert gleich aus den Beispielen ablesen?

Ja, wenn es geeignetes Ähnlichkeitsmaß zwischen Objekten/Fällen gibt.

Instanzbasiertes Lernen, Grundidee

$f(x_i) = y_j$ falls das Paar (x_j,y_j) in Beispielen enthalten und x_j das Objekt in den Beispielen ist, das x_i am ähnlichsten ist.

Zur Erhöhung der Zuverlässigkeit: betrachte die k ähnlichsten Objekte

$sim(x_i,x_j)$ liefert numerisches Maß der Ähnlichkeit zweier Objekte/Fälle

gegeben: Menge von Beispielpaaren (x_i,y_i).

Sei x neues Objekt, N_k die Menge der k Paare (x_i,y_i), für die $sim(x,x_i)$ maximal ist.

Für x prognostizierter Wert: das y, für das der folgende Ausdruck maximal wird:

$$\sum_{(xi,y) \in N_k} sim(x,x_i)$$

Beispiel

Beispiel: k = 4; die x ähnlichsten Beispiele seien

$$(x_1,a) \quad (x_2,a) \quad (x_3,a) \quad (x_4,b)$$

es gelte $sim(x,x_1) = sim(x,x_2) = sim(x,x_3) = 1$ und $sim(x,x_4) = 2$.
Damit:

$$\sum_{(x_i,a) \in N_k} sim(x,x_i) = 3$$

und

$$\sum_{(x_i,b) \in N_k} sim(x,x_i) = 2$$

Der für x prognostizierte Wert ist also a (obwohl x_4 näher an x liegt).
Für k = 1 wäre der Wert also b gewesen.

Fallbasiertes Schließen

Ähnliche Ideen liegen dem fallbasierten Schließen (case based reasoning) zugrunde:

Datenbank mit vorliegenden Fällen (jeweils Problem mit Lösung).

neuer Fall F => suche den ähnlichsten Fall F' und verwende Lösung von F'.

Gegebenenfalls muss Lösung noch an F angepasst werden.

Hauptproblem: was ist das geeignete Ähnlichkeitsmaß?

In der Praxis durchaus erfolgreiche Anwendungen.

Zusammenfassung Maschinelles Lernen

Generelles Schema des Lernens

$$B \wedge H \wedge D \models C+$$

1. Lernen von Entscheidungsbäumen

B leer, H Entscheidungsbaum, d.h. Konj. von Implikationen, D beschreibt Attribute von Objekten, keine Relationen zwischen Objekten

2. Versionsräume

B leer, nicht eine Hypothese H berechnet, sondern Menge aller Hypothesen, die obige Bedingung erfüllen

3. Induktives Logisches Programmieren

B und D definites Logikprogramm, H Menge von definiten Regeln

4. Instanzbasiertes Lernen

statt Erzeugen von Verallgemeinerungen ähnlichstes Objekt/Fall suchen und entsprechende Lösung verwenden

Data Mining/ Knowledge Discovery in Databases (KDD)

Begriffe teils synonym verwendet, teils Data Mining als ein spezieller Teilaspekt von KDD

KDD is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data (Fayyad et al.)

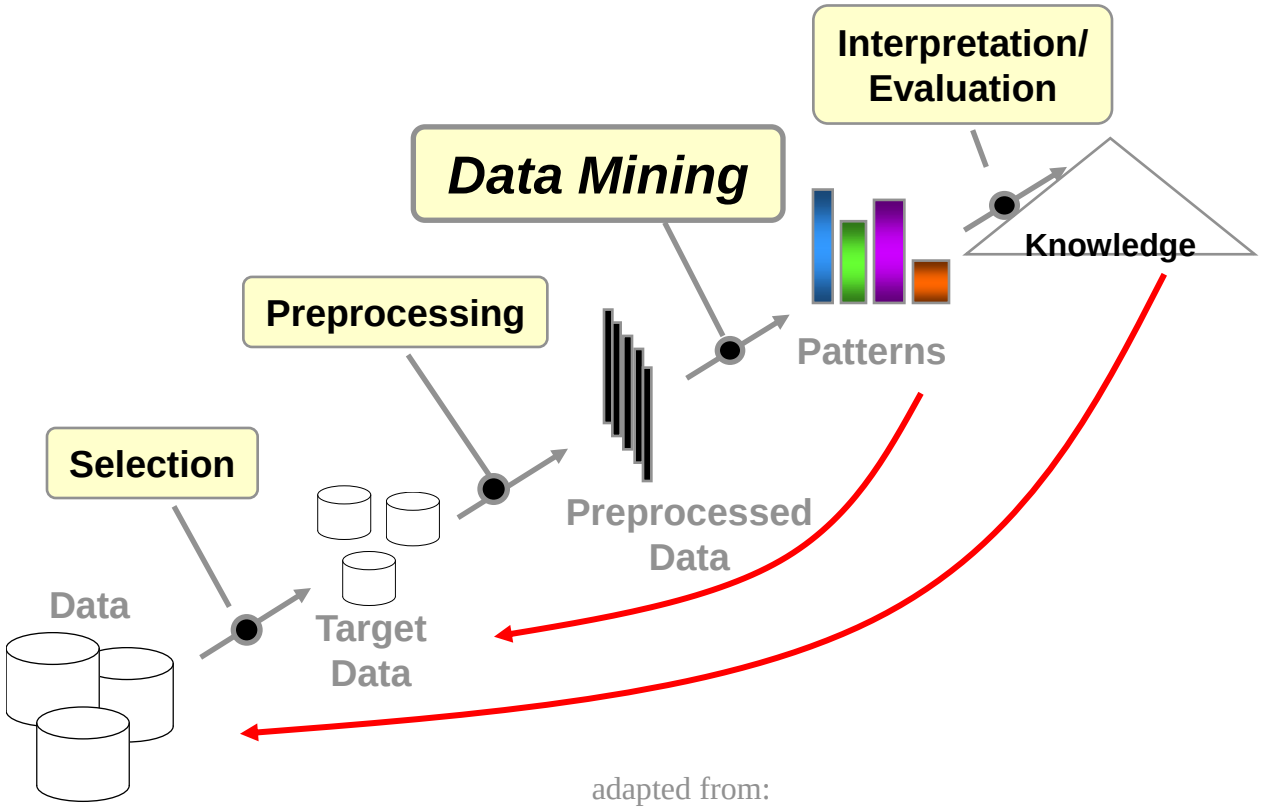
Pattern: statement in a given language that describes (relationships among) the facts in a subset of the given data and is (in some sense) simpler than the enumeration of all facts in the subset.

(Beispiele: Gleichungen, Regeln, Cluster, Entscheidungsbäume, ...)

Hauptziel: Entdecken nützlicher Information in großen Datenmengen

Gebiet etabliert seit ca. 1990

Der KDD-Prozess



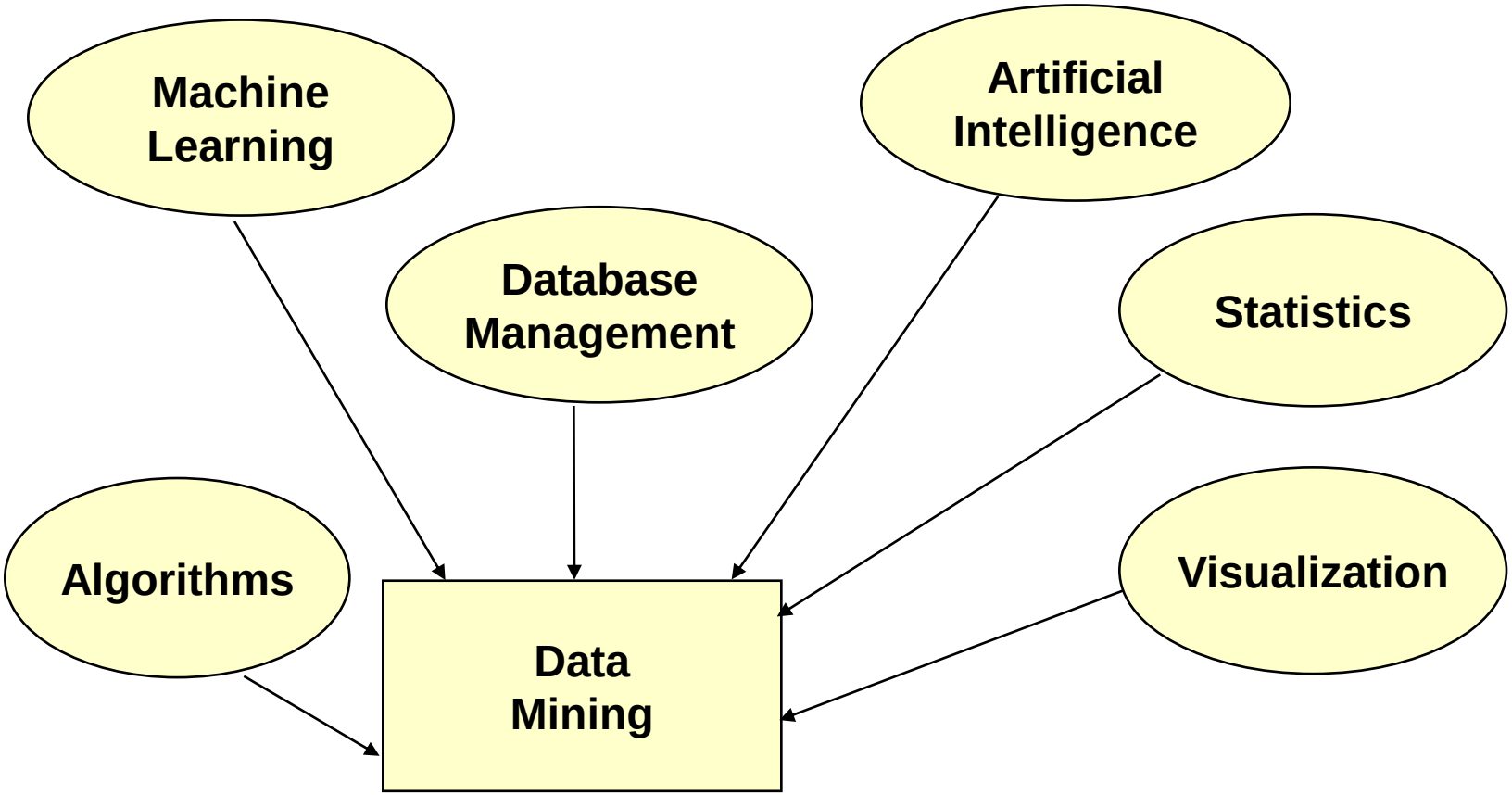
adapted from:
U. Fayyad, et al. (1995), "From Knowledge Discovery to Data Mining: An Overview," Advances in Knowledge Discovery and Data Mining, U. Fayyad et al. (Eds.), AAAI/MIT Press

Der KDD-Prozess nach Mannila

1. **Understanding the domain**
   Vorverständnis unerläßlich, vielversprechend, wenn es Experten gibt, aber Eigenschaften der Domäne sich häufig ändern (Käufergewohnheiten)
2. **Preparing the data set**
   Auswahl der Datenquellen, Integration heterogener Daten, Fehlerbehandlung, fehlende Werte etc.
3. **Discovering patterns (data mining)**
   Techniken aus Statistik und Machine Learning verwendet
4. **Postprocessing of patterns**
   Auswahl/Ordnen der gefundenen Patterns, Visualisierung, etc.
5. **Putting results to use**
   Anwendung

Lernverfahren (decision tree learning, rule induction, …) Kern des data mining.
Unterschiedliche Focussierung, hier: große Mengen einfacher Daten

KDD: Integration verschiedener Techniken



Typische Data Mining Aufgaben

Gegeben Menge von Datenbankeinträgen:

- **Klassifikation**: Vorhersage eines Attributwertes (target attribute) aus den anderen
- **Clustering**: Einteilen der Objekte in Klassen ähnlicher Objekte
- **Assoziation**: Finden von Zusammenhängen zwischen häufig vorkommenden itemsets (kein target)
- Beispiel Warenkorb: wer Brot kauft, kauft auch häufig Butter, wer Wurst und Senf kauft, kauft auch häufig Bier
- Anwendungen: Gesundheitswesen, wissenschaftliche Daten, Finanzdaten ...
- Bsp.: SKICAT (Sky Image Cataloging and Analysis Tool), 3 terabyte Bilddaten, klassifiziert 2 Milliarden Himmelsobjekte (Sterne/Galaxien), 40 Attribute pro Objekt, basiert auf decision tree learning, statistische Optimierung über mehrere gelernte Bäume, Klassifizierung 94% korrekt

Association Rule Mining

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Association Rules

{Diaper} $\rightarrow$ {Beer},
{Milk, Bread} $\rightarrow$ {Eggs,Coke},
{Beer, Bread} $\rightarrow$ {Milk},

Implication means co-occurrence, not causality!

Frequent Itemsets

- **Itemset**
  - a collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - number of occurrences of an itemset
  - E.g. σ({Milk, Bread, Diaper}) = 2
- **Support**
  - fraction of transactions that contain an itemset
  - e.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - an itemset whose support is greater than or equal to a *minsup* threshold

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Definition: Association Rule

- **Association Rule**
  - An expression of the form X => Y, where X and Y are itemsets
  - Example: {Milk, Diaper} => {Beer}

- **Rule Evaluation Metrics**
  - Support (s): fraction of transactions that contain both X and Y
  - Confidence (c): measures how often items in Y appear in transactions that contain X

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Entdecken von Assoziationsregeln in binären Daten

$R = \{A_1, ..., A_p\}$ Menge binärer Attribute

Relation $r = \{t_1, ..., t_n\}$ über Schema R: Matrix mit p Spalten und n Zeilen, jede Zeile Vektor aus 0 und 1

Beispiele: Studentendatenbank, R Kurse, n Studenten, 1 in (s,c) bedeutet Student s hat Kurs c belegt. Oder: Kunde n hat Produkt k gekauft; Publikation p zitiert Publikation p‘, ...

Assoziationsregel: $X \Rightarrow Y$, $X \subseteq R$, $Y \subseteq R \backslash X$

Intuitiv: wenn Objekt Eigenschaften in X hat, dann (meist) auch Eigenschaften Y

Gegeben $W \subseteq R$,

s(W,r) Häufigkeit von W in r: Anteil der Zeilen die für alle Attribute in W Wert 1 haben (nicht Anzahl!).

Häufigkeit einer Regel $X \Rightarrow Y$: $s(X \cup Y, r)$

Konfidenz einer Regel $X \Rightarrow Y$: $s(X \cup Y, r) / s(X, r)$

Aufgabe: finde alle Regeln $X \Rightarrow Y$ mit Häufigkeit $\geq \gamma$ und Konfidenz $\geq \theta$.

Frequent Sets

In realistischen Retail Anwendungen: ca. 10^6 Zeilen, 5000 Spalten, $\gamma = 10^{-2} - 10^{-5}$

Hunderte/Tausende von Regeln zu finden

Suchraum exponentiell in Anzahl der Attribute (beliebige Teilmengen als body)

X ist frequent set in r falls $s(X,r) \geq \gamma$.

Wenn frequent sets bestimmt, Bestimmung Assoziationsregeln einfach:

für jeden frequent set X und $Y \subseteq X$, teste ob $X \setminus Y \Rightarrow Y$ genügend hohe Konfidenz hat.

Wie findet man frequent sets? Ausnutzen der Eigenschaft: Teilmengen von frequent sets sind frequent:

> Starte mit einelementigen Mengen; finde die, die frequent sind,
> bilde daraus 2 elementige frequent sets etc., bis keine neuen mehr gefunden;
> n-elementige Menge ist Kandidat nur wenn alle ihre n-1-elementigen Teilmengen (n Stück!) frequent sind

Algorithmus zur Berechnung von frequent sets

C := {{A} | A ∈ R}; F := {}; i := 1;
While C ≠ {} **do**
    F‘ := the sets X ∈ C that are frequent;
    F := F ∪ F‘;
    i := i+1;
    C := the sets of size i such that each subset of size i-1 is in F‘
output F

Beispiel: items A, B ,C, D, E, F. {A,B,C},{C,D,E} sowie alle ihre Teilmengen seien frequent, sonst keine frequent sets:

C_1: {A}, {B}, {C}, {D}, {E}, {F}
F_1: {A}, {B}, {C}, {D}, {E}

C_2: {A, B}, {A,C}, {A,D}, {A,E}, {B,C}, {B,D}, {B,E}, {C,D}, {C,E}, {D,E}
F_2: F_1 + {A, B}, {A,C}, {B,C}, {C,D}, {C,E}, {D,E}

C_3: {A, B, C}, {C, D, E}
F_3: F_2 + {A, B, C}, {C, D, E}

C_4: leer

Anmerkungen

Wieviele mögliche Assoziationsregeln mit 1 Regelkopf gibt es im Beispiel?

=> Aus jedem mindestens 2-elementigen frequent set können alle Elemente als Regelköpfe ausgewählt werden: also 2 * 6 + 3 * 2 = 18

In typischen Fällen hat der größte frequent set bis zu 10 Elemente (damit sind mindestens alle 2^{10} Teilmengen auch frequent sets !!)

Algorithmus funktioniert in der Praxis für bis zu mehreren Dutzend Millionen Beispielen (Zeilen), falls nicht zu viele frequent sets vorkommen.

Algorithmen dürfen viel Zeit verbraten, da keinerlei Echtzeitverhalten erforderlich.

Weiteres Beispiel

	A	B	C	D	E
1	1	1	1	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	0
5	0	0	0	1	1
6	0	0	0	1	0
7	1	1	1	1	0
8	0	0	0	0	0
9	0	0	1	0	0
10	1	1	0	1	0

frequent sets mit Häufigkeit ≥ 0.4: {A}, {B}, {C}, {D}, {A,C}, {B,D}

Regeln mit Konfidenz ≥ 0.8: A => C, C => A, B => D