Frequent Pattern discovery

Hauptseminar Machine Learning WS 2003/04

Bearbeiter: Marius Morawski

Betreuer: Stefan Kramer

Einleitung

- Problemstellung:
  - Finden von Mustern/Regelmäßigkeiten in großen Datenbeständen
  - Anwendung z.B. bei der Untersuchung von Kaufverhalten im Supermarkt, Surfen im iNet, Biochemie, …
- Problem: I.d.R. große Datenbestände
  - ⇒ Durchtesten aller Möglichkeiten kommt nicht in Frage
  - Aber: die Muster sind i.d.R. voneinander abhängig!
    - Generalisierungs-/Spezialisierungs- Beziehung zwischen Mustern kann ausgenutzt werden

Ideen & Ansätze zur Vorgehensweise

- Von einfachen Mustern zu spezielleren
- Begrenzte Anzahl von Iterationen auf den Daten
- Vorraussetzung:
  - Mittel zur Repräsentation der Muster und zugrunde liegender Zusammenhänge
    - Hier: Regelbasierter Ansatz

Regeln (1)

- Einfaches Wenn- Dann- Muster:
  - Falls A, dann B (-> boolean)
- Abwandlung: Probabilistische Regeln:
  - Falls A, dann B mit Wahrscheinlichkeit P
- Bewährtes Mittel zur Wissensrepräsentation
- Einfach zu interpretieren

Regeln (2)

- Mittel zum Lernen von interpretierbarem Wissen in der Machine Learning- Forschung
  - ⇒ Classification Tree Learning
- Gute Eignung zur Darstellung von Klassifizierungen (-> boolean)
  - Erweiterbar z.B. auf Zahlenwerte (z.B. „X>25,67“)
- Boolean- Funktionen auf linker + rechter Seite meist Konjunktionen (-> einfach)

Finden von Assoziationsregeln (1)

Beispiel:

1. Mit gefordertem Häufigkeits- Grenzwert s=0,4 ergeben sich die Itemsets {A}, {B}, {C}, {D}, {AB} und {BC}.
2. Daraus könnte man z.B. die Regeln A⇒C mit c=2/3 und B⇒C mit c=1 ableiten

Basket-ID	A	B	C	D	E
t_1	1	0	0	0	0
t_2	1	1	1	1	0
t_3	1	0	1	0	1
t_4	0	0	1	0	0
t_5	0	1	1	1	0
t_6	1	1	1	0	0
t_7	1	0	1	1	0
t_8	0	1	1	0	1
t_9	1	0	0	1	0
t_{10}	0	1	1	0	1

Finden von Assoziationsregeln (2)

- Einfache und nützliche Art von Regeln für Data Mining
- Definitionen:
  - Itemset θ: Pattern von der Form $\left(A_{i_1}=1\right)\wedge\ldots\wedge\left(A_{i_k}=1\right)$
  - Assoziationsregeln: Formeln von der Form $\theta\Rightarrow\gamma$, wobei θ und γ Itemsets sind (γ besteht aber nur aus einer Klausel)
  - Häufigkeit frq(θ): Anzahl der Fälle im Datenbestand, die θ erfüllen
  - Rückhalt (~support): $frq(\theta\wedge\gamma)$
  - Konfidenz (~ accuracy): $c(\theta\Rightarrow\gamma)=\frac{fr(\theta\wedge\gamma)}{fr(\theta)}$

    (Abschätzung bedingte Wahrscheinlichkeit)

Finden von Assoziationsregeln (3)

Überlegungen:

- i.d.R. sehr große und dünn besetzte Matrizen
  - Passt i.d.R. nicht in den Speicher
    - ⇒ Ziel: möglichst wenig Durchläufe
  - Aber normalerweise nur wenige Frequent Sets
    - ⇒ Erst Frequent Sets suchen, dann aus diesen Frequent Sets die Regeln (Teilung des Problems in 2 Schritte)
- Itemsets können nur dann eine hohe Häufigkeit haben, wenn das auch für alle Subsets gilt
  - ⇒ Frequent Sets mit einer Variable bestimmen, daraus Frequent Sets der Größe 2 bilden, u.s.w.

Finden von Assoziationsregeln (4)

Generelle Vorgehensweise:

1. Finden von Frequent Itemset Patterns
   1. Minimale Häufigkeit s festlegen
   2. alle Itemset Patterns finden, die über dem Grenzwert liegen
2. Finden von Assoziationsregeln
   1. Minimale Konfidenz c festlegen
   2. Aus den gefundenen Itemset Patterns Regeln der Form $X \Rightarrow Y$ bilden, die den Konfidenzgrenzwert c erfüllen.

Finden von Assoziationsregeln (5)

- Border einer Menge S von Itemsets Bd(S)
  - Dient z.B. zur Beschreibung einer Menge von Frequent Itemsets
  - Alle Itemsets, deren sämtliche Generalisierungen zur S gehören und deren sämtliche Spezialisierungen nicht mehr zu S gehören
  - Man unterscheidet
    - $Bd^+(S)$: Der Teil von Bd(S), der noch zu S gehört
    - $Bd^-(S)$: Der Teil von Bd(S), der nicht mehr zu S gehört
    - Einer von beiden reicht

Finden von Assoziationsregeln (6)

- Beispiel für Border:
  - Itemset-Menge S = {{A}, {B}, {C}, {F}, {A, B}, {A, C}, {A, F}, {C, F}, {A, C, F}}

    mit Attributen aus {A, …, F}
  - $Bd^+(S)$={{A, B}, {A, C, F}}
  - $Bd^-(S)$={{D}, {E}, {B, C}, {B, F}}

Finden von Assoziationsregeln (7)

1. Finden von Frequent Itemset Patterns:

```
i=0; Ci={{A}|A ist Variable};
While (Ci not empty)
    Li = leere Menge von Sets der Größe i;
    For (each set S in Ci)
        If (S.isFrequent()) then Li.add(S);
    Ci+1 = leere Menge von Sets der Größe i+1;
    For (each set S in Li)
        For (each set T in Li after S)
            If (size(S U T)=i+1) then Ci+1.addEntry(S U T);
    i++;
```

Finden von Assoziationsregeln (8)

Komplexität des APriori-Algorithmus

- Kandidaten finden:

  $O(|L_i|^2)$ Paare von Sets, Verschmelzen jeweils $O(|L_i|)$

  - Im worst case von kubischer Komplexität
  - Im average case von linearer Komplexität
- Häufigkeit berechnen:

  $O(|C_i|np)$
- Für alle Schritte:

  $O(\sum_i |C_i|np)$

Finden von Assoziationsregeln (9)

Optimierungen beim Finden von Frequent Sets:

- Zahl der Durchläufe reduzieren (-> Sampling i.d.R. 2)
  1. Aus dem Sample Menge der Frequent Sets berechnen (Grenzwert dazu leicht senken)
  2. Frequenz der gefundenen Frequent Sets aus dem Sample im Gesamtdatenbestand errechnen
  3. Falls es ein Set gibt, das im Sample nicht häufig war, dessen sämtliche Subsets aber in den Gesamtdaten häufig waren, muss ein weiterer Durchlauf gestartet werden
- Zahl der möglichen Kandidaten reduzieren
- Zeit für Berechnung der Häufigkeit reduzieren
  - Daten in Baumstruktur speichern

Finden von Assoziationsregeln (10)

Alternativen/Verbesserungen zu APriori (1)

- DIC
  - Beginnt bereits während des Datenbankdurchlaufs mit dem Überprüfen eines Itemsets
- Partition
  - Durchsucht Datenbank stückweise nach Frequent Itemsets und prüft das Ergebnis dann gegen den ganzen Datenbestand
- Algorithmus mit Hashing (Park)
  - Kann einige nicht-häufige Itemsets ohne Datenbankzugriff ausschließen; Datenbank wird bei jedem Durchlauf verkleinert
- Randomisierter Algorithmus (Gunopulos)
  - Gültige Muster werden testweise erweitert bis zum Scheitern

Finden von Assoziationsregeln (11)

Alternativen/Verbesserungen zu APriori (2)

- MaxEclat/ MaxClique
  - Versuchen bereits vor der eigentlichen Suche, große Frequent Itemsets zu finden; danach APriori
- Pincer-Search
  - Versucht während der gesamten Suche, große Frequent Itemsets zu finden
- Max-Miner

Finden von Assoziationsregeln (12)

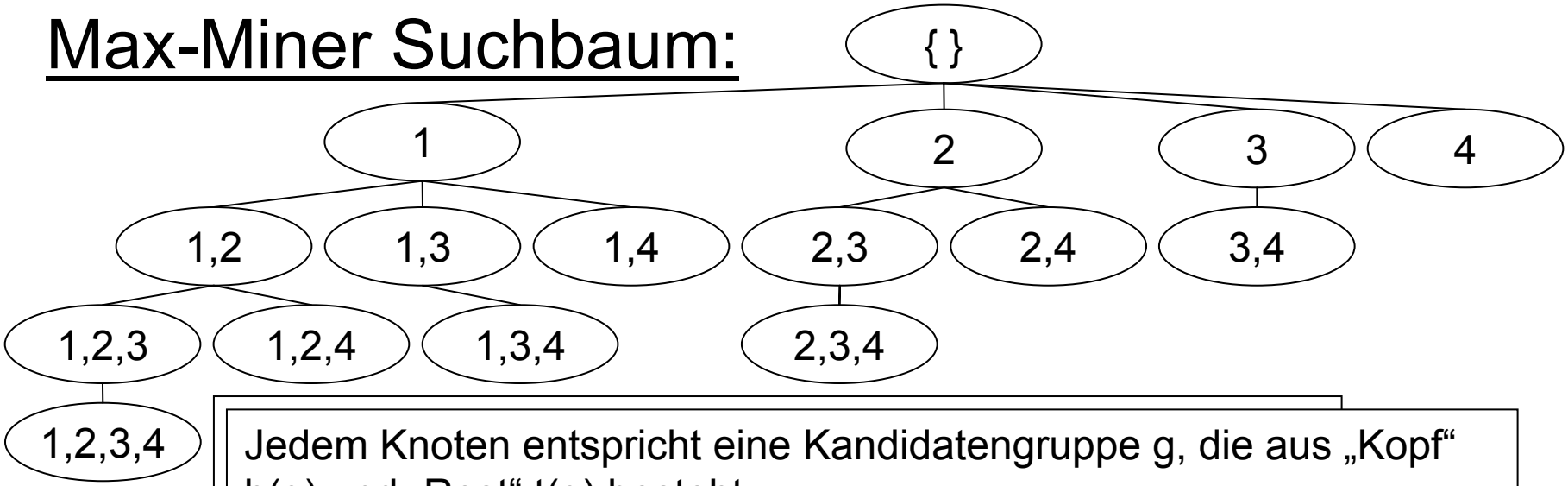
Max-Miner

- „vorausschauende“ Suche versucht immer, möglichst schnell die speziellsten (größten) Sets zu finden, die gerade noch häufig genug sind
  - Subsets davon sind automatisch auch häufig und werden nicht weiter betrachtet
- Erkennung ob Itemset häufig:
  - Kommt oft ohne weiteren Datenbankzugriff aus:

    Aus vorigen Datenbankzugriffen kann eine untere Grenze für die Häufigkeit ermittelt werden

Finden von Assoziationsregeln (13)

Max-Miner Suchbaum:



Jedem Knoten entspricht eine Kandidatengruppe g, die aus „Kopf“ h(g) und „Rest“ t(g) besteht.

Beispiel Knoten „1“: h(g)={1}, t(g)={2, 3, 4}

fr(h(g) U t(g)) > min_fr $\Rightarrow$ Alle Subsets sind häufig!

Fr(h(g) U {i $\in$ t(g)} < min_fr $\Rightarrow$ Weder Set h(g) U {i} noch Supersets davon sind häufig (wie schon bei APriori)

Finden von Assoziationsregeln (14)

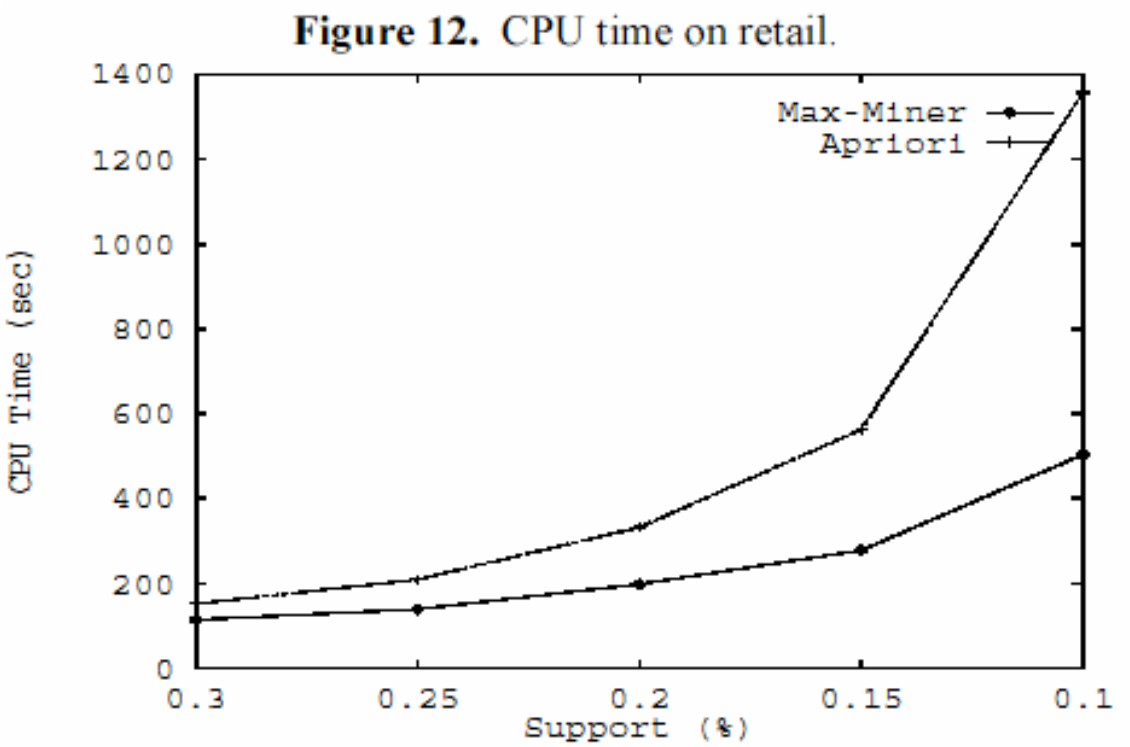
Max-Miner

- Optimierungen:
  - Finden von Maximal Frequent Itemsets fördern
    - ⇒ Baum so umsortieren, dass möglichst oft gilt: fr(h(g) U t(g)) > min_fr
    - ⇒ Häufige Items sollten in möglichst vielen Ästen auftreten
    - ⇒ Hinten einsortieren
  - Für fr(h(g) U t(g)) untere Grenze errechnen
    - Dazu reichen bereits vorher ausgelesene Informationen (→kein neuer Datenbankzugriff notwendig)

Finden von Assoziationsregeln (15)

Max-Miner VS APriori

- Beispiel Verkaufszahlen



Finden von Assoziationsregeln (16)

Assoziationsregeln generieren:

1. Aus allen gefundenen Itemsets X alle möglichen Regeln der Form $X \setminus B \Rightarrow B$ generieren
2. diejenigen auswählen, die die geforderte Treffgenauigkeit erfüllen

$\Rightarrow$ Ein Durchlauf durch den Datenbestand reicht

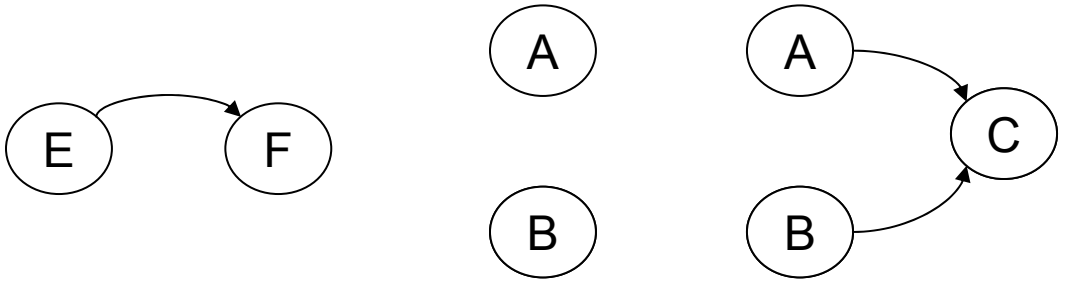
Relevanz von Assoziationsregeln

- Wie relevant sind die gefundenen Regeln?
  - Mögliche Kriterien:
    - Häufigkeit frq? Ja, bereits abgedeckt
    - Konfidenz c? Nicht unbedingt!
    - Statistische Signifikanz?
      - $P(B=1|A=1) \approx P(B=1)$? Dann ist $A \Rightarrow B$ uninteressant.
      - Aber: nicht immer sehr aussagekräftig

Verallgemeinerung auf andere Problemstellungen

- Voraussetzungen:
  - Als Konjunktion darstellbar (A_1^A_2^…^A_n)
  - Monoton
  - Möglichkeit, eine Reihe schnell auf ein Muster zu testen -> Häufigkeitsberechnung
  - Nicht zu viele Frequent Patterns
- Beispiele:
  - Finding Episodes In Sequences
  - Mustersuche nach anderen Kriterien als (nur) Häufigkeit

Finding Episodes In Sequences (1)



- Problemstellung:
  - Finde in einer Ereignissequenz S in allen Fenstern der Größe w alle Episoden aus einer Menge ε von Episoden, die mit einer Häufigkeit von mindestens min_fr auftreten.
- Idee:
  - analog zum Finden von Frequent Patterns erst einfache Muster suchen und darauf aufbauend komplexere Muster

Finding Episodes In Sequences (2)

- Definitionen:
  - *Ereignissequenz S*: Folge von Paaren (e, t)
    - e Ereignistyp
    - t Zeit des Auftretens
  - *Episode α*: Partielle Ordnung auf Ereignistypen
  - *Fensterbreite w*: Breite eines Ausschnitts von Ereignissen aus einer Ereignissequenz S
  - *Häufigkeit fr(α):* der Anteil der Ausschnitte mit Länge w aus S, der die Episode α enthält
  - *Subepisode:* Es gilt $\beta \prec \alpha$, falls alle Knoten und Relationen aus β auch in α auftreten und *β≠α*.

Finding Episodes In Sequences (3)

- Pseudo-Code

```
L=1; C_1= alle Episoden a der Länge 1;
While (C_l not empty)
    L_l = alle Episoden a aus C_l mit fr(a, s, win) >= min_fr
    l = l + 1
    Cl = alle Episoden a der Länge l, deren sämtliche
    Subepisoden β in L_|β| enthalten sind
```

Finding Episodes In Sequences (3)

Generalized Episodes

- Ereignisse können (weitere) Attribute haben
  - Z.B. `x.page = y.page`
- Flexiblere Modellierung der zeitlichen Abhängigkeiten
  - Z.B. `x.page = p1 ^ y.page = p2 [30]`
    - ⇒ `z.page = p3 [60]`
  - ⇒ Konfidenz für verschiedene kann leichter für mehrere Zeitgrenzen berechnet werden

Finding Episodes In Sequences (4)

Generalized Episodes

- Ereignis e = (a_1, …, a_n, t)
- Episode P = Konjunktion von Prädikaten ψ über Ereignisse
- Prädikat ψ
  - x.t <= y.t oder
  - x.a (z.B. Testen auf Gleichheit mit Konstante) oder
  - x.a ∘ y.b (z.B. Testen auf Gleichheit)

  (x, y sind Ereignisse mit Attributen a und b)
- Episodenregel: P [V] $\Rightarrow$ Q[W]
  - P, Q sind Episoden
  - V und W sind Zahlenwerte (Zeit)

Finding Episodes In Sequences (5)

Generalized Episodes

- Idee für Finden von Generalized Episodes:
  1. küzeste Ausschnitte des Intervalls finden, die die jeweiligen Episoden enthalten, finden („Minimal Occurrences“)
  2. Daraus Kandidaten für größere häufige Episoden generieren

  ⇒ Wieder ähnlich APriori
- Problem: Finden von beliebigen Episoden P in einer Folge von Ereignissen S ist NP-vollständig
  - Aber: „einfache“ Episoden (nur einstellige Prädikate) kann in Zeit |P||S| erfolgen.
  - Und: Auch beliebige Episoden können in der Regel schnell gefunden werden.

Mustersuche nach allgemeineren Kriterien (1)

- Motivation:
  - Oft will man die Suche weiter einschränken oder die Häufigkeit ist uninteressant
  - Beispiel:

    Die folgenden Regeln mit den Variablen Alter und Einkommen wurden entdeckt:

    α: Age>40 $\Rightarrow$ Einkommen>62755 (Wahrscheinlichkeit 0,34)

    β: Age>41 $\Rightarrow$ Einkommen>62855 (Wahrscheinlichkeit 0,33)

    Aussage fast identisch, außerdem existieren zu jedem Muster zu viele Spezialisierungen, sodass der Aufwand zu hoch wird.

Mustersuche nach allgemeineren Kriterien (2)

- Algorithmus zur Frequent Pattern Discovery verallgemeinerten Kriterien:
  - Kriterium Interessantheit:

    Bestimmt, ob das Muster ausgegeben wird
  - Kriterium Potential:

    Bestimmt, ob Spezialisierungen von diesem Muster potential interessant sein könnten (unabhängig von Interessantheit!)
  - Die beiden Kriterien können beispielsweise Häufigkeit, Konfidenz oder andere sein.

Mustersuche nach allgemeineren Kriterien (3)

```
C = allgemeinstes Muster;
While (C not empty)
   E = alle geeigneten Spezialisierungen der Elemente von
      C;
   For (each q in E)
      If (q erfüllt Interessantheits- Kriterium)
         Gib q als ein Ergebnis aus
      If (q erfüllt Potential- Kriterium NICHT)
         lösche q aus E;
   Filtere E nach weiteren Kriterien;
   C = E;
```

Mustersuche nach allgemeineren Kriterien (4)

- Konkrete Umsetzungen:
  - Frequent Pattern Discovery:
    - Muster sind Itemsets
    - Definiere Interessantheit und Potential als die Häufigkeit eines Itemsets
  - Suche nach signifikantesten Regeln:
    - Muster sind von der Form $\alpha 1 \wedge \ldots \wedge \alpha n \Rightarrow \beta$, α und β sind Bedingungen von der Form $X=c$, $X<c$ oder $X>c$
    - Interessantheit = statistische Signifikanz
    - Potential = wahr
    - Weiteres Filtern = nimm die K Regeln mit den höchsten Signifikanz- Werten

Interessantheits- Kriterien

- Kriterien ergeben sich aus Hintergrundwissen
- Statistische Kriterien
  - Unabhängig vom Anwendungsgebiet
  - Es hat sich herausgestellt, dass die meisten sinnvollen statistischen Maße für Interessantheit zu ähnlichen Ergebnissen führen
    - Sinnvoll heißt dabei soviel wie dass bestimmte Voraussetzungen erfüllt werden sollten (z.B. sollte ein Muster, das bei gleicher Treffgenauigkeit häufiger auftritt als ein anderes, höher bewertet werden)

Statistische Maße für Interessantheit (1)

- Kontingenztabelle:
  - Stellt dar, wie häufig die rechte Seite θ und die linke Seite φ abhängig voneinander auftreten.
  - Daraus lassen sich verschiedene Maße den Zusammenhang zwischen θ und φ berechnen, z.B. das J-Maß.

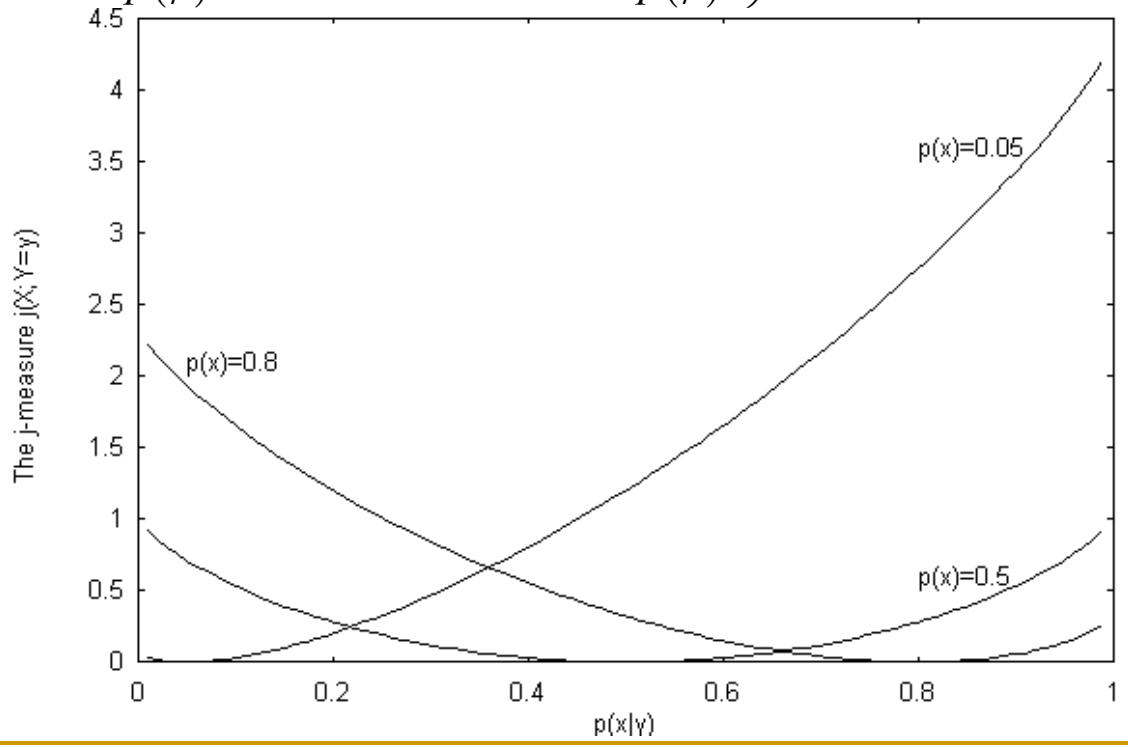
	φ	~φ
θ	fr(θ ∧ φ)	fr(θ ∧ ~φ)
~θ	fr(~θ ∧ φ)	fr(~θ ∧ ~ φ)

Statistische Maße für Interessantheit (2)

- Das J-Maß

$$J(\theta \Rightarrow \varphi) = p(\theta)\left(p(\varphi \mid \theta)\log\frac{p(\varphi \mid \theta)}{p(\varphi)} + (1 - p(\varphi \mid \theta))\log\frac{1 - p(\varphi \mid \theta)}{1 - p(\varphi)}\right)$$

$p(\varphi \mid \theta)$ ist dabei die beobachtete Konfidenz der Regel. $p(\varphi)$ und $p(\theta)$ sind die aus Beobachtung abgeschätzten Wahrscheinlichkeiten für φ und θ



Globales Modell aus lokalen Regeln?

- Möglichkeit 1: Klassifizierung per Regelsatz
  - Sortierte Liste von Regeln der Form $\theta_i \Rightarrow B=b_i$ (θ_i ist Muster, b_i ist möglicher Wert von B)
  - Alle möglichen Sortierungen nach der optimalen Sortierung zu durchsuchen wäre zu aufwendig
  - $\Rightarrow$ Als „Weighted Set Cover Problem“ auffassen
  - $\Rightarrow$ Greedy- Suche bringt relativ gute Annäherung

Globales Modell aus lokalen Regeln?

- Möglichkeit 2: Wahrscheinlichkeitsverteilung
  - Idee: Beobachtete Häufigkeiten spiegeln Wahrscheinlichkeitsverteilung wider
  1. Beginne mit Zufallsverteilung über den Variablen
  2. Passe die Verteilung an die Häufigkeits- Verteilung aller Muster θ_i an.
     1. Summiere p über alle Zustände, für die θ_i wahr ist.
     2. Skaliere diese Wahrscheinlichkeiten so, dass beim neuen p θ_i in $fr(\theta_i)$ Fällen wahr wird.
     - Wiederhole diese Schritte abwechselnd für alle θ_i, bis die beobachteten Häufigkeiten mit den von p vorhergesagten zusammenpassen.

Résumée Frequent Pattern Discovery

- Nur bei dünn besetzten Matrizen mit vertretbarem Aufwand durchführbar
- Liefert Zusammenhänge statt Klassifizierung
- Regeln der Form $A \Rightarrow B$ gut verständlich, weil sie der menschlichen Denkweise entgegenkommen