Übersicht

- Grundlagen für Assoziationsregeln
- Apriori Algorithmus
- Verschiedene Datenformate
- Finden von Assoziationsregeln mit mehren unteren Schranken für Unterstützung
- Finden von Assoziationsregeln für Klassifikation
- Finden von sequenziellen Mustern
- Zusammenfassung

Finden von sequenziellen Mustern

- Assoziationregeln berücksichtigen nicht die Reihenfolge der Transaktionen.
- Bei vielen Anwendungen ist die Reihenfolge wichtig,
  - Warenkorbanalyse, es ist interessant ob Artikel hintereinander gekauft werden,
    - z.B. erst der Bett kaufen und dann das Bettzeug.
  - Bei der Analyse von Web Log-Daten ist die Reihenfolge wichtig in der ein Anwender die Web-Seiten besucht hat

Grundkonzepte (1/2)

- Sei $I = \{i_1, i_2, \ldots, i_m\}$ eine Menge von Artikeln.
- **Sequenz:** Eine geordnete Liste von Artikelmengen.
- **Artikelmenge/Element**: Eine nicht-leere Menge von Artikeln $X \subseteq I$. Eine Sequenz s wird geschrieben als $\langle a_1 a_2 \ldots a_r \rangle$, wobei a_i eine Artikelmenge ist, die auch **Element** von s genannt wird.
- Ein Element (oder eine Artikelmenge) einer Sequenz wird beschrieben durch $\{x_1, x_2, \ldots, x_k\}$, wobei $x_j \in I$ ein Artikel ist.
- Ohne Beschränkung der Allgemeinheit nehmen wir an, dass die Artikel eines Elements einer Sequenz in **lexikographischer Ordnung** sind.

Grundkonzepte (2/2)

- **Größe**: Die **Größe** einer Sequenz ist die Anzahl der Elemente (oder Artikelmengen) in der Sequenz.
- **Länge**: Die **Länge** einer Sequenz ist die Anzahl der Artikel in der Sequenz.
  - Eine Sequenz der Länge *k* ist ene ***k*-Sequenz**.
- Eine Sequenz $s_1 = \langle a_1 a_2 \ldots a_r \rangle$ ist eine **Teilsequenz** einer anderen Sequenz $s_2 = \langle b_1 b_2 \ldots b_v \rangle$,
  oder s_2 ist eine **Obersequenz** von s_1, falls es folgende Indizes gibt $1 \leq j_1 < j_2 < \ldots < j_{r-1} < j_r \leq v$, s.d.
  $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}$, …, $a_r \subseteq b_{jr}$.
  s_2 enthält s_1.

Beispiel

- Sei $I = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- Die Sequenz $\langle\{3\}\{4, 5\}\{8\}\rangle$ ist in $\langle\{6\}\ \{3, 7\}\{9\}\{4, 5, 8\}\{3, 8\}\rangle$ enthalten
  - weil $\{3\} \subseteq \{3, 7\}$, $\{4, 5\} \subseteq \{4, 5, 8\}$, und $\{8\} \subseteq \{3, 8\}$.
  - Jedoch, $\langle\{3\}\{8\}\rangle$ ist nicht in $\langle\{3, 8\}\rangle$ enthalten, die Umkehrung gilt auch nicht.
  - Die Größe der Sequenz $\langle\{3\}\{4, 5\}\{8\}\rangle$ ist 3, und die Länge der Sequenz ist 4.

Ziel

- Gegeben sei eine Menge *S* mit Sequenzen, dann ist das Problem des Findens aller sequentiellen Muster alle Sequenzen zu finden, deren Unterstützung größer als eine vorgegebene untere Schranke ist.
- Alle solche Sequenzen sind **häufige Sequenzen**, oder **sequentielle Muster**.
- Die Unterstützung einer Sequenz ist der Anteil der Sequenzen aus S welche diese enthalten

Beispiel (1/2)

Table 1. A set of transactions sorted by customer ID and transaction time

Customer ID	Transaction Time	Transaction (items bought)
1	July 20, 2005	30
1	July 25, 2005	90
2	July 9, 2005	10, 20
2	July 14, 2005	30
2	July 20, 2005	40, 60, 70
3	July 25, 2005	30, 50, 70
4	July 25, 2005	30
4	July 29, 2005	40, 70
4	August 2, 2005	90
5	July 12, 2005	90

Beispiel (2/2)

Table 2. Data sequences produced from the transaction database in Table 1.

Customer ID	Data Sequence
1	⟨{30} {90}⟩
2	⟨{10, 20} {30} {40, 60, 70}⟩
3	⟨{30, 50, 70}⟩
4	⟨{30} {40, 70} {90}⟩
5	⟨{90}⟩

Table 3. The final output sequential patterns

	Sequential Patterns with Support ≥ 25%
1-sequences	⟨{30}⟩, ⟨{40}⟩, ⟨{70}⟩, ⟨{90}⟩
2-sequences	⟨{30} {40}⟩, ⟨{30} {70}⟩, ⟨{30} {90}⟩, ⟨{40, 70}⟩
3-sequences	⟨{30} {40, 70}⟩

GSP mining Algorithmus

- Sehr ähnlich zum Apriori Algorithmus

Algorithm GSP(S)

```
1   C_1 ← init-pass(S);                          // the first pass over S
2   F_1 ← {⟨{f}⟩ | f ∈ C_1, f.count/n ≥ minsup}; // n is the number of sequences in S
3   for (k = 2; F_{k-1} ≠ ∅; k++) do             // subsequent passes over S
4     C_k ← candidate-gen-SPM(F_{k-1});
5     for each data sequence s ∈ S do            // scan the data once
6       for each candidate c ∈ C_k do
7         if c is contained in s then
8           c.count++;                           // increment the support count
9       end
10    end
11    F_k ← {c ∈ C_k | c.count/n ≥ minsup}
12  end
13  return ⋃_k F_k;
```

Fig. 12. The GSP Algorithm for generating sequential patterns

Candidate generation

Function candidate-gen-SPM(F_{k-1})

1. **Join step.** Candidate sequences are generated by joining F_{k-1} with F_{k-1}. A sequence s_1 joins with s_2 if the subsequence obtained by dropping the first item of s_1 is the same as the subsequence obtained by dropping the last item of s_2. The candidate sequence generated by joining s_1 with s_2 is the sequence s_1 extended with the last item in s_2. There are two cases:
   - the added item forms a separate element if it was a separate element in s_2, and is appended at the end of s_1 in the merged sequence, and
   - the added item is part of the last element of s_1 in the merged sequence otherwise.

   When joining F_1 with F_1, we need to add the item in s_2 both as part of an itemset and as a separate element. That is, joining $\langle \{x\} \rangle$ with $\langle \{y\} \rangle$ gives us both $\langle \{x, y\} \rangle$ and $\langle \{x\}\{y\} \rangle$. Note that x and y in $\{x, y\}$ are ordered.
2. **Prune step**. A candidate sequence is pruned if any one of its $(k-1)$-subsequence is infrequent (without minimum support).

Fig. 13. The candidate-gen-SPM() function

An example

Table 4. Candidate generation: an example

Frequent 3-sequences	Candidate 4-sequences	
	after joining	after pruning
⟨{1, 2} {4}⟩	⟨{1, 2} {4, 5}⟩	⟨{1, 2} {4, 5}⟩
⟨{1, 2} {5}⟩	⟨{1, 2} {4} {6}⟩	
⟨{1} {4, 5}⟩		
⟨{1, 4} {6}⟩		
⟨{2} {4, 5}⟩		
⟨{2} {4} {6}⟩		

Übersicht

- Grundlagen für Assoziationsregeln
- Apriori Algorithmus
- Verschiedene Datenformate
- Finden von Assoziationsregeln mit mehren unteren Schranken für Unterstützung
- Finden von Assoziationsregeln für Klassifikation
- Finden von sequenziellen Mustern
- Zusammenfassung

Zusammenfassung

- Finden von Assoziationsregeln wurde sehr intensiv beforscht.
- Ebenso das Finden von sequentiellen Mustern
- Es gibt sehr viele effiziente Algorithmen und Modellvariationen.
- Verwandte Arbeiten sind
  - Regeln mit Oberklassen von Artikeln,
  - Finden von Regeln mit Zusatzbedingungen
  - Inkrementelles Finden von Regeln
  - Maximale häufig Artikelmengen
  - Geschlossene Artikelmengen
  - Interessantheit und Visualisierung von Regeln
  - Parallele Algorithmen
  - …