Sequential pattern mining

- Association rule mining does not consider the order of transactions.
- In many applications such orderings are significant. E.g.,
  - In market basket analysis, it is interesting to know whether people buy some items in sequence,
    - e.g., buying bed first and then bed sheets some time later.
  - In Web usage mining, it is useful to find navigational patterns of users in a Web site from sequences of page visits of users

Sequential pattern mining

Introduction

Sequence Data

4 customers

The transactions are ordered chronologically

Customer	A	B	C	D	E	F
C1	1	1	0	1	1	1
C2	0	0	1	1	0	1
C1	0	1	1	0	1	1
C3	0	0	0	1	1	1
C2	1	1	1	1	0	1
C1	0	1	0	0	1	0
C3	1	0	1	1	1	0
C2	0	1	0	0	1	0
C4	1	1	1	1	1	1
C1	0	0	1	1	1	1
C4	0	1	0	0	0	1
C3	1	0	1	1	1	1
C2	0	1	1	0	0	0
C4	1	0	1	1	1	1
C2	0	1	0	0	0	0

Sequential pattern mining

Introduction

Event: In Sequence Mining terminology a transaction is called an *Event*

Sequence: A *sequence* is an ordered list of events

Customer	A	B	C	D	E	F
C1	1	1	0	1	1	1
C1	0	1	1	0	1	1
C1	0	1	0	0	1	0
C1	0	0	1	1	1	1

Sequential pattern mining

Introduction

A sequence α is denoted as ($\alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_q$) where α_i is an event

Customer	A	B	C	D	E	F
C1	1	1	0	1	1	1
C1	0	1	1	0	1	1
C1	0	1	0	0	1	0
C1	0	0	1	1	1	1

Sequential pattern mining

Introduction

Sub-Sequence

It is a sequence within the sequence, preserving that order

Its events may not be adjacent, but their ordering should not violate the ordering of the bigger sequence

A subsequence can be obtained from a sequence by deleting some items and/or events

Sequential pattern mining

Introduction

Frequency & Frequent Sequence

The *frequency* of a sequence is the total number of input sequences that support it

A *frequent* sequence is a sequence whose frequency exceeds some user-specified threshold

A frequent sequence is *maximal* if it is not a sub-sequence of another frequent sequence

Sequential pattern mining

Introduction

The sequences of all customers can be written in the following form

Customer	A	B	C	D	E	F
C1	1	1	0	1	1	1
C1	0	1	1	0	1	1
C1	0	1	0	0	1	0
C1	0	0	1	1	1	1

Sequence 1: (A, B, D, E, F) → (B, C, E, F) → (B, E) → (C, D, E, F)
Sequence 2: (C, D, F) → (A, B, C, D, F) → (B, E) → (B, C) → (B)
Sequence 3: (D, E, F) → (A, C, D, E) → (A, C, D, E, F)
Sequence 4: (A, B, C, D, E, F) → (B, F) → (A, C, D, E, F)

Sequential pattern mining

Examples:

(A, C) is a sub-sequence of sequence 2, 3 & 4, but not of 1

The sub-sequence (A) → (C) is a subsequence of sequence 1

The sub-sequence (A) → (C), means that item C appears in a transaction after (not necessarily, immediately after) another transaction containing A

Sequence 1: (A, B, D, E, F) → (B, C, E, F) → (B, E) → (C, D, E, F)
Sequence 2: (C, D, F) → (A, B, C, D, F) → (B, E) → (B, C) → (B)
Sequence 3: (D, E, F) → (A, C, D, E) → (A, C, D, E, F)
Sequence 4: (A, B, C, D, E, F) → (B, F) → (A, C, D, E, F)

Sequential pattern mining

Examples:

The frequency of (A, C) is 3, because we do not count multiple occurrences of (A, C) in the same sequence

The total number of sequences that support (A, C) are 3 namely sequence 2, 3 and 4

Sequence 1: (A, B, D, E, F) → (B, C, E, F) → (B, E) → (C, D, E, F)
Sequence 2: (C, D, F) → (A, B, C, D, F) → (B, E) → (B, C) → (B)
Sequence 3: (D, E, F) → (A, C, D, E) → (A, C, D, E, F)
Sequence 4: (A, B, C, D, E, F) → (B, F) → (A, C, D, E, F)

Sequential pattern mining

Examples:

The frequency of (B) → (D) is 2, as sequence 1 and 4 support it

Note that sequence 4 supports (B, E) → (B) but it does not supports (B) → (B, E)

Sequence 1: (A, B, D, E, F) → (B, C, E, F) → (B, E) → (C, D, E, F)
Sequence 2: (C, D, F) → (A, B, C, D, F) → (B, E) → (B, C) → (B)
Sequence 3: (D, E, F) → (A, C, D, E) → (A, C, D, E, F)
Sequence 4: (A, B, C, D, E, F) → (B, F) → (A, C, D, E, F)

Sequential pattern mining

Introduction

Normally, a sequence mining problem is concerned with the temporal order of events within a sequence

However, we can also focus on the temporal distance between the events. That is (A) → (C) should indicate the temporal gap between the transactions containing A and the transaction containing C

Period of times are relevant because most relationships between two events ceases to be effective after some period of time (e.g. consuming a beverage and stomach upset)

Sequential pattern mining

Introduction

Thus, we can specify the time distance in terms of a distance threshold, *d*

So, (B) $\rightarrow_d$ (B, E) denotes the event (B, E) occurs within the distance of *d* transactions after the transaction containing B

Sequential pattern mining

Introduction

Example:
There is no sequence which supports (A, B, D) $\rightarrow_1$ D
However, (A, B, D) $\rightarrow_2$ D is supported by sequence 4

Sequence 1: (A, B, D, E, F) → (B, C, E, F) → (B, E) → (C, D, E, F)
Sequence 2: (C, D, F) → (A, B, C, D, F) → (B, E) → (B, C) → (B)
Sequence 3: (D, E, F) → (A, C, D, E) → (A, C, D, E, F)
Sequence 4: (A, B, C, D, E, F) → (B, F) → (A, C, D, E, F)

Sequential pattern mining

Introduction

- Let $I = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- Sequence $\langle\{3\}\{4, 5\}\{8\}\rangle$ is **contained** in (or is a **subsequence** of) $\langle\{6\}\ \{3, 7\}\{9\}\{4, 5, 8\}\{3, 8\}\rangle$
- because $\{3\} \subseteq \{3, 7\}$, $\{4, 5\} \subseteq \{4, 5, 8\}$, and $\{8\} \subseteq \{3, 8\}$.
- However, $\langle\{3\}\{8\}\rangle$ is not contained in $\langle\{3, 8\}\rangle$ or vice versa.
- The size of the sequence $\langle\{3\}\{4, 5\}\{8\}\rangle$ is 3, and the length of the sequence is 4.

Example

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

Example (cond)

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}⟩

Sequential pattern mining

GSP Algorithm

The algorithms for solving sequence mining problems are mostly based on the A-priori algorithm

One such algorithm is GSP algorithm

It is exactly like A-priori algorithm and makes multiple passes over the database

In the first pass, all 1-sequences are counted. From the frequent 1-sequences a set of candidate 2-sequences are formed, and another pass is made to gather their support

Sequential pattern mining

GSP Algorithm

The frequent 2-sequences are used to generate the candidate 3-sequences. Pruning is done among the candidates to eliminate any sequence, at least one of whose sub-sequence is not frequent

The generation of candidate sequences are as follows:

For the first pass all the items are considered as 1-sequence

Suppose after the first pass the 1-sequences (A), (B) & (C) are found to be frequent

Sequential pattern mining

GSP Algorithm

The following 2-sequences would be generated

(A) → (B)
(B) → (A)
(A) → (C)
(C) → (A)
(B) → (C)
(C) → (B)
(AB)
(AC)
(BC)

Sequential pattern mining

GSP Algorithm

Suppose after the next pass, the following 2-sequences are found to be frequent

(A) → (B)
(B) → (C)

From these 2-sequences, the following 3 sequence would be generated

(A) → (B) → (C)

GSP mining algorithm

Very similar to the Apriori algorithm

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

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).

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}⟩		