Sequential Pattern Mining

PicsArt Analytics Department

ALBERT SARGSYAN
ERIK HAMBARDZUMYAN

Business background

From theory to pattern

- Testing specific sequences based on business knowledge(funnel analytics)

2) From all available patterns to theory

- Using data mining algorithms to get all strong patterns and extract business knowledge.

Initial Purpose

The sequence mining task is to discover a set of patterns, shared across time among a large number of objects in a given database

Basic concepts

Let I = $\{i_1, i_2, \ldots, i_m\}$ be a set of m distinct items. An ***event*** is a non-empty unordered collection of items.

A ***sequence*** is an ordered list of events. An event is denoted as (i_1 i_2 ...i_k), where i_j is an item. A sequence α is denoted as ($\alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_q$)

If the event α_i occurs before α_j, we denote it as $\alpha_i < \alpha_j$. We say α is a *subsequence* of another sequence β, denoted as $\alpha \preccurlyeq \beta$, if there exist integers $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ such that $\alpha_1 \subseteq \beta_{j1}$, $\alpha_2 \subseteq \beta_{j2}, \ldots, \alpha_n \subseteq \beta_{jn}$

e.g (B → AC) is a subsequence of (AB → E → ACD), since

B ⊆ AB and AC ⊆ ACD

The ***support*** or ***frequency*** of a sequence, denoted σ (α, D), is the the total number of input-sequences in the database D that contain α.

Given a user-specified threshold called the *minimum support* (denoted *min sup*), we say that a sequence is *frequent* if it occurs more than *min sup* times.

The downward property

Apriori: Any subsequence of a frequent itemset must be frequent. Thus, If there is any itemset which is infrequent, its supersequence should not even be generated.

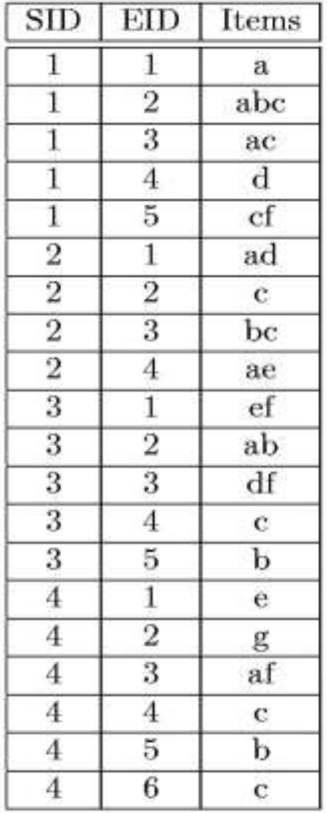
The SPADE Algorithm

SPADE (Sequential PAttern Discovery using Equivalent Class) developed by Zaki 2001

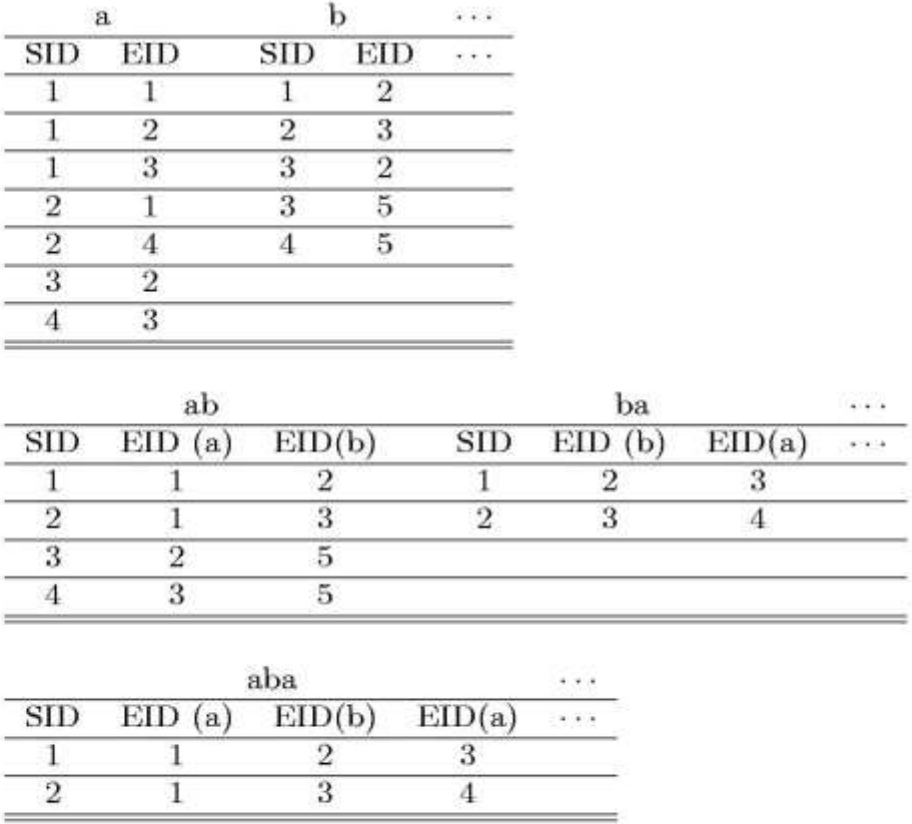
A vertical format sequential pattern mining method

A sequence database is mapped to a large set of Item: <SID, EID>. Sequential pattern mining is performed by growing the subsequences (patterns) one item at a time by *Apriori* candidate generation

The SPADE Algorithm



SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c



a		b		. . .
SID	EID	SID	EID	. . .
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			

ab			ba			. . .
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)	. . .
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

aba				. . .
SID	EID (a)	EID(b)	EID(a)	. . .
1	1	2	3	
2	1	3	4	

Challenges on Sequential Pattern Mining

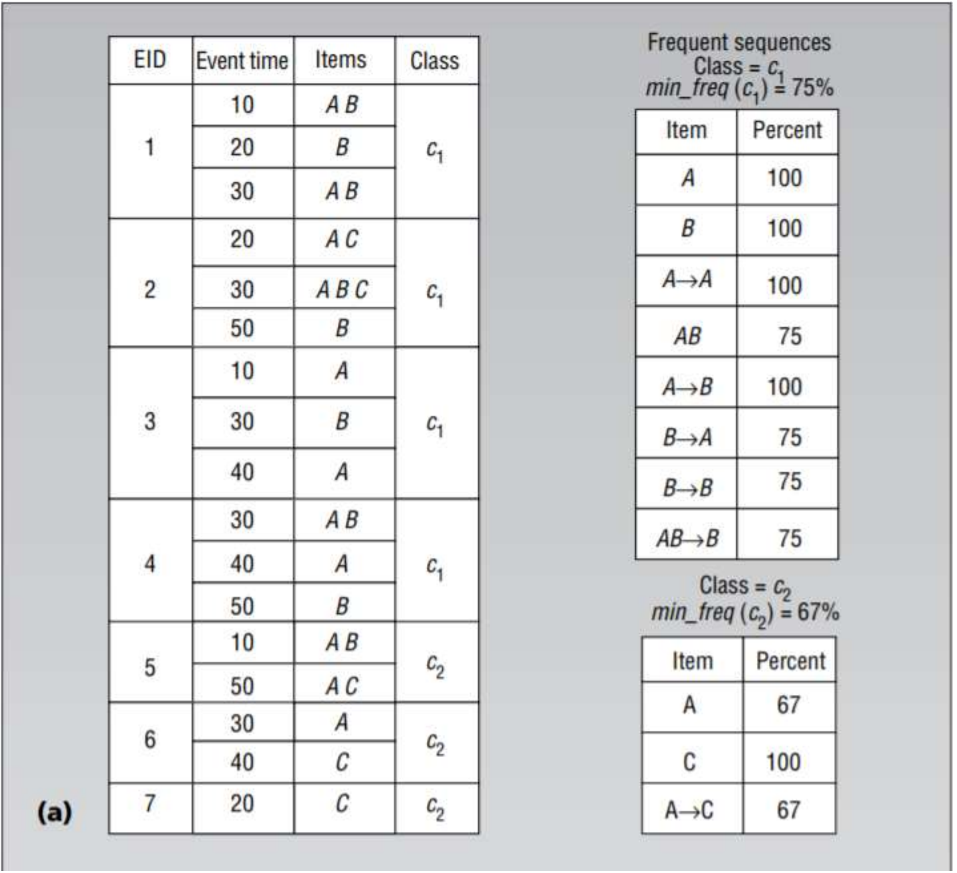
1. Obtaining a representative dataset for exploring user behavior.
2. Effective pruning of redundant sequences
3. Business interpretation

FeatureMine algortihm

FeatureMine combines sequence mining and classification algorithms to efficiently select features from large data sets.

It enables to convert the patterns discovered by the mining algorithm into a set of boolean features to feed into standard classification algorithms.

Feature enumeration



EID	Event time	Items	Class
1	10	*A B*	c_1
	20	*B*	
	30	*A B*	
2	20	*A C*	c_1
	30	*A B C*	
	50	*B*	
3	10	*A*	c_1
	30	*B*	
	40	*A*	
4	30	*A B*	c_1
	40	*A*	
	50	*B*	
5	10	*A B*	c_2
	50	*A C*	
6	30	*A*	c_2
	40	*C*	
7	20	*C*	c_2

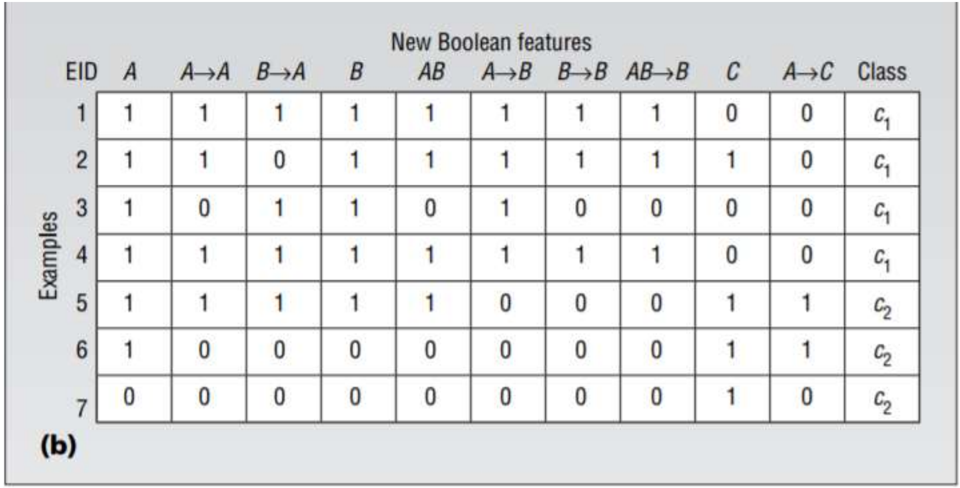
Frequent sequences
Class = c_1
min_freq (c_1) = 75%

Item	Percent
A	100
B	100
A→*A*	100
AB	75
A→*B*	100
B→*A*	75
B→*B*	75
AB→*B*	75

Class = c_2
min_freq (c_2) = 67%

Item	Percent
A	67
C	100
A→C	67

(a)



New Boolean features

Examples: EID	*A*	*A*→*A*	*B*→*A*	*B*	*AB*	*A*→*B*	*B*→*B*	*AB*→*B*	*C*	*A*→*C*	Class
1	1	1	1	1	1	1	1	1	0	0	c_1
2	1	1	0	1	1	1	1	1	1	0	c_1
3	1	0	1	1	0	1	0	0	0	0	c_1
4	1	1	1	1	1	1	1	1	0	0	c_1
5	1	1	1	1	1	0	0	0	1	1	c_2
6	1	0	0	0	0	0	0	0	1	1	c_2
7	0	0	0	0	0	0	0	0	1	0	c_2

(b)

Selection criteria for mining

1. Features should be frequent.

2. Features should be distinctive of at least one class.

3. Feature sets should not contain redundant features.

Pruning of rules

The third criteria implies two pruning rules

$$conf(\beta, c, \mathcal{D}) = \frac{fr(\beta, \mathcal{D}_c)}{fr(\beta, \mathcal{D})}$$

Lemma 1: If $f_i < f_j$ and conf(f_i , c, D) = 1.0, then f_i subsumes f_j with respect to class c.

Lemma 2: Let $a = a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow$ an where A, B ∈ a_i for some 1 ≤ i ≤ n. If A B, then a will be subsumed by $a_1 \rightarrow \ldots a_{i-1} \rightarrow (a_i - B) \rightarrow$ $a_{i+1} \ldots \rightarrow a_n$.

Our novelties

Overfrequency threshold: a user specified threshold Ω that removes features that have very high frequency;

thus, do not have any business value as they are very intuitive. Removing those features will significantly reduce computation.

over_freq := σ (α, D)> Ω

Our novelties

Interclass threshold: threshold ε specifies maximum gap between supports of two classes.

We do not need to keep those features from two classes where relative supports are close since as they are not distinctive for either class1 or class2.

```
SELECT * from class1 JOIN class2 WHERE
class1.features=class2.features AND
ABS(class1.sup-class2.sup)<=ε
```