Frequent subsequence mining

Robert Kessl

SUI, 18. March 2010

Outline

Frequent substructure mining

- We have a database $\mathcal{D}$ of transactions t.
- t can be an arbitrary object.
- For example: itemsets (basket market), time sequences, graphs
- Mining of frequent substructures has exponential complexity (in the worst case)

Frequent subsequence mining

- We denote the set of all items by $\mathcal{I} = \{b_i\}$. We impose some ordering on the items in the set $\mathcal{I}$, i.e., $b_1 < b_2 < \ldots < b_{|\mathcal{I}|}$
- We denote the set of all events by $\mathcal{E} = \mathcal{P}(\mathcal{I})$
- Let $\alpha_i \in \mathcal{E}, 1 \leq i \leq n$ be an event.
- A sequence is an ordered list: $\alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_n$, e.g., $\mathcal{I} = \{A, B, C, D, E, F\}$, $A \rightarrow AB \rightarrow BCD \rightarrow E$

Notation: a sequence $\clubsuit$ contains events $\clubsuit_i$, i.e., $\clubsuit_1 \rightarrow \clubsuit_2 \rightarrow \ldots \rightarrow \clubsuit_n$.

Subsequence

Definition (subseqence)

Let have two sequences $\alpha = \alpha_1 \rightarrow \ldots \rightarrow \alpha_n$ and $\beta = \beta_1 \rightarrow \ldots \rightarrow \beta_m, m \leq n$. We call β the subsequence of α, denoted by $\beta \preceq \alpha$ iff there exists one-to-one order preserving function $f : \alpha \rightarrow \beta$ that maps events in β to events in α, that is:

1. $\alpha_i \subseteq \beta_l = f(\alpha_i)$
2. if $\alpha_i < \alpha_j$ then $f(\alpha_i) < f(\alpha_j)$, i.e., $\beta_k = f(\alpha_i), \beta_l = f(\alpha_j)$ such that $\beta_k < \beta_l$

Some subsequences of $A \rightarrow AB \rightarrow BCD \rightarrow E$:

- $A \rightarrow A$
- $A \rightarrow E$
- $AB \rightarrow B \rightarrow E$
- $\cancel{AE}$

Problem formulation

Database $\mathcal{D}$:

TID	Transaction
1	$A \to AB \to BCD \to E$
2	$CE \to AB \to F \to CDE$
3	$BE \to B \to AF \to ACE$
4	$A \to E \to BF$
5	$BCD \to AF \to ABF$

- we are searching for subsequence in the transactions $t \in \mathcal{D}$ that occurs in at least *min_support* transactions.

Problem formulation

Database $\mathcal{D}$:

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

- we are searching for subsequence in the transactions $t \in \mathcal{D}$ that occurs in at least *min_support* transactions.
- for example, the sequence $A \rightarrow A$ occurs in 3 transactions.

Prefix and suffix of a sequence

Let have three sequences: $\begin{array}{l}\alpha = \alpha_1 \rightarrow \ldots \rightarrow \alpha_n, \\ \beta = \beta_1 \rightarrow \ldots \rightarrow \beta_m, m < n, \\ \gamma = \gamma_1 \rightarrow \ldots \rightarrow \gamma_k, k \leq n.\end{array}$

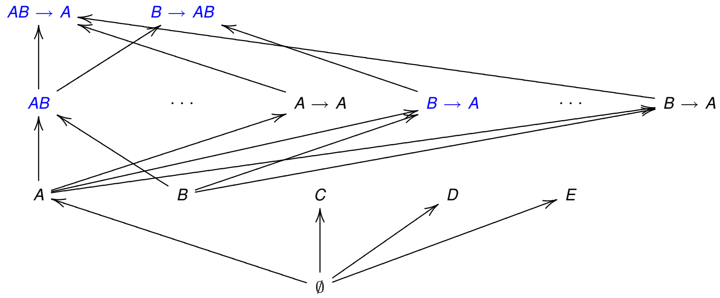
$$\begin{array}{ccccccc} \alpha_1 & \ldots & \alpha_{m-1} & \alpha_m & \alpha_{m+1} & \ldots & \alpha_n \\ \beta_1 & \ldots & \beta_{m-1} & \beta_m \cup \gamma_1 & \gamma_2 & \ldots & \gamma_k \end{array}$$

Then β is the prefix and γ is the suffix of α.
Denoted by $\alpha = \beta.\gamma$ or $\gamma = \alpha \setminus \beta$
Example, given a sequence $AB \rightarrow AF \rightarrow BCD$:

1. prefix A, suffix $\_B \rightarrow AF \rightarrow BCD$.
2. prefix AB, suffix $AF \rightarrow BCD$.

The hyperlattice

Part of the lattice of all sequences L:



- top $\top$ of the lattice L is $\top = \infty$.
- bottom $\perp$ of the lattice L is an empty sequence $\emptyset$
- Let α, β be two sequences, then:
  - Meet of α, β is the set of minimal uppper bounds, denoted by $\alpha \wedge \beta$.
  - Join of α, β is the set of all maximal lower bounds, denoted by $\alpha \vee \beta$.

The Prefix-Based Equivalence Classes

- DFS algorithms partitions the hyperlattice into smaller

Definition

Let α be a sequence. The prefix-based equivalence class, denoted by $[\alpha]$ is the set of all sequences having α as a prefix.

The prefix-based equivalence class is a sub-hyperlattice of L.

Generating sequences

Generating sequences: let P be an arbitrary sequence and $a, b, c, d \in \mathcal{I}$. We can combine sequences $P \rightarrow a$, $P \rightarrow b$, Pc, Pd in the following ways:

1. $P \rightarrow a \rightarrow b$
2. $P \rightarrow b \rightarrow a$
3. $P \rightarrow ab$
4. $P \rightarrow a \rightarrow a$
5. Pcd
6. $Pc \rightarrow a$
7. $Pc \rightarrow b$
8. $\ldots$

Generating sequences

Generating sequences: let P be an arbitrary sequence and $a, b, c, d \in \mathcal{I}$. We can combine sequences $P \rightarrow a$, $P \rightarrow b$, Pc, Pd in the following ways:

1. $P \rightarrow a \rightarrow b$
2. $P \rightarrow b \rightarrow a$
3. $P \rightarrow ab$
4. $P \rightarrow a \rightarrow a$
5. Pcd
6. $Pc \rightarrow a$
7. $Pc \rightarrow b$
8. $\ldots$

We must order the operations !!

The monotonicity of support

Lemma (Monotonicity of support)

Let α be a sequence with support Supp$(\alpha, \mathcal{D})$ in database $\mathcal{D}$. For every superset β of α ($\alpha \preceq \beta$) holds: Supp$(\alpha, \mathcal{D}) \geq$ Supp$(\beta, \mathcal{D})$.

$A \rightarrow A$

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

The monotonicity of support

Lemma (Monotonicity of support)

Let α be a sequence with support Supp$(\alpha, \mathcal{D})$ in database $\mathcal{D}$. For every superset β of α ($\alpha \preceq \beta$) holds: Supp$(\alpha, \mathcal{D}) \geq$ Supp$(\beta, \mathcal{D})$.

$A \rightarrow AB$

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

The monotonicity of support

Lemma (Monotonicity of support)

Let α be a sequence with support Supp$(\alpha, \mathcal{D})$ in database $\mathcal{D}$. For every superset β of α ($\alpha \preceq \beta$) holds: Supp$(\alpha, \mathcal{D}) \geq$ Supp$(\beta, \mathcal{D})$.

$A \rightarrow ABF$

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

Abstract substructure mining

- A database $\mathcal{D}$, a language $\mathcal{L}$;
- sentences $\varphi, \Phi \in \mathcal{L}$;
- a frequency criterion $q(\varphi) \in \{\text{true}, \text{false}\}$;
- a monotone specialization/generalization relation: $\varphi \preceq \Phi$
- $q(\Phi) = \text{true} \Rightarrow q(\varphi) = \text{true}$

Generalization of the Apriori algorithm

$C_1 \leftarrow \{\varphi \in \mathcal{L} | \text{there is no } \varphi' \text{ such that } \varphi' \prec \varphi\}$
$i \leftarrow 1$
while C_i not empty **do**
    $F_i \leftarrow \{\varphi \in C_i | q(\varphi) = \mathit{true}\}$
    $C_{i+1} \leftarrow \{\varphi \in \mathcal{L} | \forall \varphi' \prec \varphi \text{ we have } \varphi' \in \bigcup_{j \leq i} F_j\} \setminus \bigcup_{j \leq i} C_j$
    $i \leftarrow i + 1$
end while
return $F_1 \cup F_2 \cup \ldots \cup F_{k-1}$

Algorithms

- The GSP algorithm: an Apriory like algorithm
- The Spade algorithm: DFS algorithm that uses TID lists
- The PrefixSpan algorithm: DFS algorithm that uses *projected database*

The GSP algorithm

- BFS algorithm.
- *Generate&test* approach.
- Let α be the longest sequence in $\mathcal{D}$ with length k, denoted by $|\alpha| = k$. **The GSP algorithm can make k scans of $\mathcal{D}$**

A *candidate* sequence $\alpha, |\alpha| = k$:

- Support of α is unknown.
- all $\beta \preceq \alpha, |\beta| = k - 1$ are frequent, i.e., $Supp(\beta) \geq min\_support$.

The GSP algorithm contd.

GSP(**In**: Database $\mathcal{D}$,**In**: Integer *min_supp*, **In**/**Out**: Set F)

$\mathcal{F}_1 \leftarrow$ {frequent 1-sequences}
for $k \leftarrow 2; \mathcal{F}_{k-1} \neq 0; k \leftarrow k+1$ **do**
    $\mathcal{F}_k \leftarrow \emptyset$
    $C_k \leftarrow$ candidates created from $\mathcal{F}_{k-1}$
    for all $\beta \in C_k$ **do**
        $\beta.support \leftarrow$ support of β in $\mathcal{D}$
        if $\beta.support \geq min\_supp$ **then**
            $\mathcal{F}_k \leftarrow \mathcal{F}_k \bigcup \beta$
        end if
    end for
    $F \leftarrow F \bigcup \mathcal{F}_k$
end for

The Spade algorithm

1. DFS algorithm.
2. Uses TID lists.
3. Similar algorithm as the Eclat algorithm.
4. Created by the author of the Eclat algorithm (M.J. Zaki).

TID lists

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

TID	EID	Event
1	1	A
1	2	AB
1	3	BCD
1	4	E
2	1	CE
2	2	AB
2	3	F
2	4	CDE
3	1	BE
3	2	B
3	3	AF
3	4	ACE
4	1	A
4	2	E
4	3	BF
5	1	BCD
5	2	AF
5	3	ABF

TID lists contd.

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

A's TID list

TID	EID	Event
1	1	A
1	2	AB
2	2	AB
3	3	AF
3	4	ACE
4	1	A
5	2	AF
5	3	ABF

TID lists contd.

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

B's TID list

TID	EID	Event
1	2	AB
1	3	BCD
2	2	AB
3	1	BE
3	2	B
4	3	BF
5	1	BCD
5	3	ABF

TID lists contd.

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

C's TID list

TID	EID	Event
1	3	BCD
2	1	CE
2	4	CDE
3	4	ACE
5	1	BCD

TID lists contd.

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

D's TID list

TID	EID	Event
1	3	BCD
2	4	CDE
5	1	BCD

TID lists contd.

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

E's TID list

TID	EID	Event
1	4	E
2	1	CE
2	4	CDE
3	1	BE
3	4	ACE
4	2	E

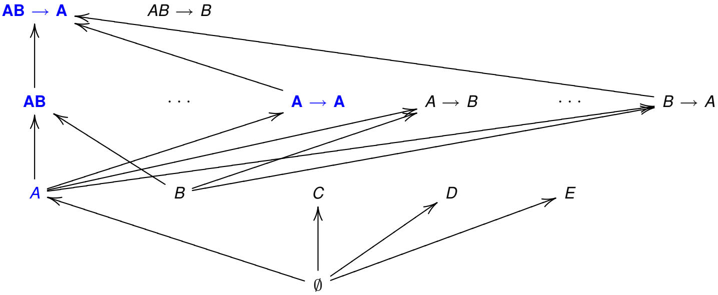
TID lists contd.

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

F's TID list

TID	EID	Event
2	3	F
3	3	AF
4	3	BF
5	2	AF
5	3	ABF

The hyperlattice



Temporal TID list join

Example:

A's TID list

1	1	A
1	2	AB
2	2	AB
3	3	AF
3	4	ACE
4	1	A
5	2	AF
5	3	ABF

B's TID list

1	2	AB
1	3	BCD
2	2	AB
3	1	BE
3	2	B
4	3	BF
5	1	BCD
5	3	ABF

$A \rightarrow B$' TID list

1	2	AB
1	3	BCD
4	3	BF
5	3	ABF

Temporal TID list join

Example:

A's TID list

1	1	A
1	2	AB
2	2	AB
3	3	AF
3	4	ACE
4	1	A
5	2	AF
5	3	ABF

B's TID list

1	2	AB
1	3	BCD
2	2	AB
3	1	BE
3	2	B
4	3	BF
5	1	BCD
5	3	ABF

$B \rightarrow A$'s TID list

3	3	AF
3	4	ACE
5	2	AF
5	3	ABF

Temporal TID list join

Example:

A's TID list

1	1	A
1	2	AB
2	2	AB
3	3	AF
3	4	ACE
4	1	A
5	2	AF
5	3	ABF

B's TID list

1	2	AB
1	3	BCD
2	2	AB
3	1	BE
3	2	B
4	3	BF
5	1	BCD
5	3	ABF

AB's TID list

1	2	AB
2	2	AB

The Spade algorithm

SPADE(**In**: AtomSet ϵ,**In**: Integer min_supp, **In**/**Out**: Set $\mathcal{F}$)

for all atoms $A_i \in \epsilon$ **do**
    $T_i \leftarrow \{\}$
    for all atoms $A_j \in \epsilon, j \geq i$ and all combinations α of A_i, A_j **do**
        $\mathcal{L}(\alpha) =$ temporal TID list join of $\mathcal{L}(A_i)$ with $\mathcal{L}(A_j)$
        if $Supp(\alpha) \geq min\_supp$ **then**
            $T_i \leftarrow T_i \bigcup \{\alpha\}$
            $F = F \bigcup \alpha$
        end if
    end for
    Spade(T_i, min_supp, $\mathcal{F}$)
end for

The PrefixSpan algorithm

1. DFS algorithm.
2. Uses database projection.
3. Pattern-growth algorithm
4. Reduced candidate generation.
5. Created by the author of the FPGrowth algorithm (J. Han).

Database Projection

Collecting of suffixes projected from sequences by following a given prefix.

Definition (Sequence projection)

Let α, β, γ be three sequences. We say that γ is α-*projected* sequence in β iff $\alpha.\gamma$ is a maximal subsequence of β, denoted by $\beta|_\alpha$.

$\beta = (A \rightarrow B \rightarrow A \rightarrow B \rightarrow AC \rightarrow D)$
$\alpha = (A \rightarrow B)$
α-*projected* sequence in β, i.e., $\beta|_\alpha$, is $\gamma = (A \rightarrow B \rightarrow AC \rightarrow D)$.
$\beta = (A \rightarrow BC \rightarrow B \rightarrow AC) \Rightarrow \beta|_\alpha = (\_C \rightarrow B \rightarrow AC)$

Database Projection example

$\mathcal{D}$ - a database we project from

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

$\xrightarrow{\alpha=(AB)}$

$\mathcal{D}|_\alpha$ - α-projected database

TID	Transaction
1	$BCD \rightarrow E$
2	$F \rightarrow CDE$
5	$\_F$

$\Rightarrow$ Support of C ?

Database Projection example

$\mathcal{D}$ - a database we project from

TID	Transaction
1	$A \rightarrow AB \rightarrow BCD \rightarrow E$
2	$CE \rightarrow AB \rightarrow F \rightarrow CDE$
3	$BE \rightarrow B \rightarrow AF \rightarrow ACE$
4	$A \rightarrow E \rightarrow BF$
5	$BCD \rightarrow AF \rightarrow ABF$

$\overset{\alpha=(AB)}{\Longrightarrow}$

$\mathcal{D}|_\alpha$ - α-projected database

TID	Transaction
1	$BCD \rightarrow E$
2	$F \rightarrow CDE$
5	$\_F$

$\Rightarrow$ Support $Supp(AB \rightarrow C, \mathcal{D}) = Supp(C, \mathcal{D}|_\alpha)$

Prefixspan Pseudocode

PREFIXSPAN-RECURSIVE(**In**: Database $\mathcal{D}_\alpha$, **In**: Sequence α, **In**: Integer min_supp, **In/Out**: Set $\mathcal{F}$)

$\mathcal{F}_1 \leftarrow$ {frequent items in $\mathcal{D}_\alpha$}
for all items $b_i \in \mathcal{F}_1$ **do**
  $\beta = (\alpha_1 \rightarrow \cdots \rightarrow (\alpha_n \bigcup \{b_i\}))$
  $\gamma = (\alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow (b_i))$
  if *Supp*$(\beta, \mathcal{D}_\alpha) \geq$ min_supp **then**
    $\mathcal{F} \leftarrow \mathcal{F} \bigcup \{\beta\}$
    $\mathcal{D}' \leftarrow (\mathcal{D}_\alpha)|_\beta$
    Prefixspan-Recursive($\mathcal{D}'$, β, min_supp, $\mathcal{F}$)
  end if
  if *Supp*$(\gamma, \mathcal{D}_\alpha) \geq$ min_supp **then**
    $\mathcal{F} \leftarrow \mathcal{F} \bigcup \{\gamma\}$
    $\mathcal{D}' \leftarrow (\mathcal{D}_\alpha)|_\gamma$
    Prefixspan-Recursive($\mathcal{D}'$, γ, min_supp, $\mathcal{F}$)
  end if
end for

Mining sequential patterns with constraints

- *Event time* – let $T : \mathcal{I} \rightarrow \mathbf{R}$, the function t assignes timestamp to each event in the sequence.
- For each sequence α it holds that $T(\alpha_i) < T(\alpha_j), i < j$.

Let α, β, be two sequences such that α is subsequence of β. A constraint C is:

- Anti-monotonic: iff $C(\beta)$ implies $C(\alpha)$
- Monotonic: iff $C(\alpha)$ implies $C(\beta)$

Timing constraints – the maxspan/minspan

Maxspan/Minspan: the maximum/minimum allowed time difference between the latest and earliest occurances of events in α in the transaction t:

$$t = A \rightarrow AB \rightarrow BCD \rightarrow E$$

- maxspan=2, supports: $A \rightarrow A$, $A \rightarrow B$, $A \rightarrow BC$.
- maxspan=2, does not supports: $A \rightarrow E$.
- minspan=2, does not supports: $A \rightarrow A$, $A \rightarrow B$, $A \rightarrow BC$.
- minspan=2, supports: $A \rightarrow E$.
- the maxspan is anti-monotonic.
- the minspan is monotonic.

Mingap/Maxgap

Mingap/Maxgap: is the *minimum/maximum* time difference of occurences of events from α in a transaction *t*.

$$t = A \rightarrow AB \rightarrow BCD \rightarrow E$$

- mingap=2, *t* supports: $A \rightarrow E$.
- mingap=2, *t* does not supports: $A \rightarrow A$.
- maxgap=1, *t* supports: $A \rightarrow C$.
- maxgap=1, *t* does not supports: $A \rightarrow E$.
- mingap/maxgap is anti-monotnic.

Regular expressions

- Regular expression: each regular expression $\mathcal{R}$ can be represented by a finite state automaton.
- Each event in the sequence α must contain exactly one item.
- A frequent sequence α is valid if it matches a state of the finite state automaton representing $\mathcal{R}$.