Frequent Pattern Mining

Albert Bifet (@abifet)

Paris, 18 October 2016

albert.bifet@telecom-paristech.fr

Association Rules

Dataset Example

Id	Items
1	bread, butter, milk
2	eggs, milk, yogurt
3	bread, cheese, eggs, milk
4	eggs, milk, yogurt
5	cheese, milk, yogurt

Association rule: **eggs, milk** $\rightarrow$ **yogurt**

$$conf(X \rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)}$$

Frequent Patterns

Suppose $\mathscr{D}$ is a dataset of patterns, $t \in \mathscr{D}$, and *min_sup* is a constant.

Definition
Support (t): number of patterns in $\mathscr{D}$ that are superpatterns of t.

Definition
Pattern t is *frequent* if *Support* (t) $\geq$ *min_sup*.

Frequent Subpattern Problem
Given $\mathscr{D}$ and *min_sup*, find all frequent subpatterns of patterns in $\mathscr{D}$.

Frequent Patterns

Suppose $\mathscr{D}$ is a dataset of patterns, $t \in \mathscr{D}$, and *min_sup* is a constant.

Definition

Support (t): number of patterns in $\mathscr{D}$ that are superpatterns of t.

Definition

Pattern t is *frequent* if *Support* $(t) \geq$ *min_sup*.

Frequent Subpattern Problem

Given $\mathscr{D}$ and *min_sup*, find all frequent subpatterns of patterns in $\mathscr{D}$.

Frequent Patterns

Suppose $\mathscr{D}$ is a dataset of patterns, $t \in \mathscr{D}$, and *min_sup* is a constant.

Definition

Support (t): number of patterns in $\mathscr{D}$ that are superpatterns of t.

Definition

Pattern t is *frequent* if *Support* $(t) \geq$ *min_sup*.

Frequent Subpattern Problem

Given $\mathscr{D}$ and *min_sup*, find all frequent subpatterns of patterns in $\mathscr{D}$.

Frequent Patterns

Suppose $\mathscr{D}$ is a dataset of patterns, $t \in \mathscr{D}$, and *min_sup* is a constant.

Definition

Support (t): number of patterns in $\mathscr{D}$ that are superpatterns of t.

Definition

Pattern t is *frequent* if *Support* $(t) \geq$ *min_sup*.

Frequent Subpattern Problem

Given $\mathscr{D}$ and *min_sup*, find all frequent subpatterns of patterns in $\mathscr{D}$.

Pattern Mining

Dataset Example

Document	Patterns
d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	Frequent
d1,d2,d3,d4,d5,d6	c
d1,d2,d3,d4,d5	e,ce
d1,d3,d4,d5	a,ac,ae,ace
d1,d3,d5,d6	b,bc
d2,d4,d5,d6	d,cd
d1,d3,d5	ab,abc,abe be,bce,abce
d2,d4,d5	de,cde

minimal support = 3

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	**Frequent**
6	c
5	e,ce
4	a,ac,ae,ace
4	b,bc
4	d,cd
3	ab,abc,abe be,bce,abce
3	de,cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	**Frequent**	**Gen**	**Closed**
6	c	c	c
5	e,ce	e	ce
4	a,ac,ae,ace	a	ace
4	b,bc	b	bc
4	d,cd	d	cd
3	ab,abc,abe	ab	
	be,bce,abce	be	abce
3	de,cde	de	cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	Frequent	Gen	Closed	Max
6	c	c	c	
5	e,ce	e	ce	
4	a,ac,ae,ace	a	ace	
4	b,bc	b	bc	
4	d,cd	d	cd	
3	ab,abc,abe be,bce,abce	ab be	abce	abce
3	de,cde	de	cde	cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	**Frequent**	**Gen**	**Closed**	**Max**
6	c	c	c	
5	e,ce	e	ce	
4	a,ac,ae,ace	a	ace	
4	b,bc	b	bc	
4	d,cd	d	cd	
3	ab,abc,abe be,bce,abce	ab be	abce	abce
3	de,cde	de	cde	cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

$e \longrightarrow ce$

Support	Frequent	Gen	Closed	Max
6	c	c	c	
5	e,ce	e	ce	
4	a,ac,ae,ace	a	ace	
4	b,bc	b	bc	
4	d,cd	d	cd	
3	ab,abc,abe	ab		
	be,bce,abce	be	abce	abce
3	de,cde	de	cde	cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	Frequent	Gen	Closed	Max
6	c	c	c	
5	e,ce	e	ce	
4	a,ac,ae,ace	a	ace	
4	b,bc	b	bc	
4	d,cd	d	cd	
3	ab,abc,abe	ab		
	be,bce,abce	be	abce	abce
3	de,cde	de	cde	cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	Frequent	Gen	Closed	Max
6	c	c	c	
5	e,ce	e	ce	
4	a,ac,ae,ace	a	ace	
4	b,bc	b	bc	
4	d,cd	d	cd	
3	ab,abc,abe	ab		
	be,bce,abce	be	abce	abce
3	de,cde	de	cde	cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

$a \rightarrow ace$

Support	Frequent	Gen	Closed	Max
6	c	c	c	
5	e,ce	e	ce	
4	a,ac,ae,ace	a	ace	
4	b,bc	b	bc	
4	d,cd	d	cd	
3	ab,abc,abe be,bce,abce	ab be	abce	abce
3	de,cde	de	cde	cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	**Frequent**	**Gen**	**Closed**	**Max**
6	c	c	c	
5	e,ce	e	ce	
4	a,ac,ae,ace	a	ace	
4	b,bc	b	bc	
4	d,cd	d	cd	
3	ab,abc,abe	ab		
	be,bce,abce	be	abce	abce
3	de,cde	de	cde	cde

Closed Patterns

Usually, there are too many frequent patterns. We can compute a smaller set, while keeping the same information.

Example

A set of 1000 items, has $2^{1000} \approx 10^{301}$ subsets, that is more than the number of atoms in the universe $\approx 10^{79}$

Closed Patterns

A priori property

If t' is a subpattern of t, then $Support\ (t') \geq Support\ (t)$.

Definition

A frequent pattern t is *closed* if none of its proper superpatterns has the same support as it has.

Frequent subpatterns and their supports can be generated from closed patterns.

Maximal Patterns

Definition
A frequent pattern t is *maximal* if none of its proper superpatterns is frequent.

Frequent subpatterns can be generated from maximal patterns, but not with their support.

All maximal patterns are closed, but not all closed patterns are maximal.

Non streaming frequent itemset miners

Representation:

- Horizontal layout

T1: a, b, c
T2: b, c, e
T3: b, d, e

- Vertical layout

a: 1 0 0
b: 1 1 1
c: 1 1 0

Search:

- Breadth-first (levelwise): Apriori
- Depth-first: Eclat, FP-Growth

The Apriori Algorithm

APRIORI ALGORITHM

Initialize the item set size $k = 1$
Start with single element sets
Prune the non-frequent ones
while there are frequent item sets
    do create candidates with one item more
        Prune the non-frequent ones
        Increment the item set size $k = k + 1$

Output: the frequent item sets

The Eclat Algorithm

Depth-First Search

- divide-and-conquer scheme : the problem is processed by splitting it into smaller subproblems, which are then processed recursively
    - **conditional database for the prefix a**
        - transactions that contain a
    - **conditional database for item sets without a**
        - transactions that not contain a
- Vertical representation
- Support counting is done by intersecting lists of transaction identifiers

The FP-Growth Algorithm

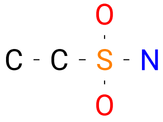
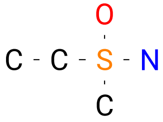
Depth-First Search

- divide-and-conquer scheme : the problem is processed by splitting it into smaller subproblems, which are then processed recursively
  - **conditional database for the prefix a**
    - transactions that contain a
  - **conditional database for item sets without a**
    - transactions that not contain a
- Vertical and Horizontal representation : FP-Tree
  - prefix tree with links between nodes that correspond to the same item
- Support counting is done using FP-Tree

Mining Graph Data

Problem

Given a data set $\mathscr{D}$ of graphs, find frequent graphs.

Transaction Id	Graph
1	
2	
3	

The gSpan Algorithm

gSpan(g, D, min_sup, S)

Input: A graph g, a graph dataset D, min_sup.
Output: The frequent graph set S.

if $g \neq min(g)$
    then return S
insert g into S
update support counter structure
$C \leftarrow \emptyset$
for each g' that can be *right-most* extended from g in one step
    do if support$(g) \geq min_sup$
        then insert g' into C
for each g' in C
    do $S \leftarrow$ gSpan(g', D, min_sup, S)
return S