

Chapter 2:

Association Rules & Sequential Patterns

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

Association rule mining

- Proposed by **Agrawal et al in 1993**.
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are categorical.
- No good algorithm for numeric data.
- Initially used for **Market Basket Analysis** to find how items purchased by customers are related.

Bread \rightarrow Milk [sup = 5%, conf = 100%]

The model: data

- $I = \{i_1, i_2, \dots, i_m\}$: a set of *items*.
- Transaction t :
 - t a set of items, and $t \subseteq I$.
- Transaction Database T : a set of transactions
 $T = \{t_1, t_2, \dots, t_n\}$.

Transaction data: supermarket data

- Market basket transactions:

t1: {bread, cheese, milk}

t2: {apple, eggs, salt, yogurt}

...

...

tn: {biscuit, eggs, milk}

- Concepts:

- *An item*: an item/article in a basket
- *I*: the set of all items sold in the store
- *A transaction*: items purchased in a basket; it may have TID (transaction ID)
- *A transactional dataset*: A set of transactions

Transaction data: a set of documents

- **A text document data set. Each document is treated as a “bag” of keywords**

doc1: Student, Teach, School

doc2: Student, School

doc3: Teach, School, City, Game

doc4: Baseball, Basketball

doc5: Basketball, Player, Spectator

doc6: Baseball, Coach, Game, Team

doc7: Basketball, Team, City, Game

The model: rules

- A transaction t contains X , a set of items (itemset) in I , if $X \subseteq t$.
- An association rule is an implication of the form:

$$X \rightarrow Y, \text{ where } X, Y \subset I, \text{ and } X \cap Y = \emptyset$$

- An itemset is a set of items.
 - E.g., $X = \{\text{milk, bread, cereal}\}$ is an itemset.
- A k -itemset is an itemset with k items.
 - E.g., $\{\text{milk, bread, cereal}\}$ is a 3-itemset

Rule strength measures

- **Support:** The rule holds with **support** sup in T (the transaction data set) if $sup\%$ of transactions contain $X \cup Y$.
 - $sup = \Pr(X \cup Y)$.
- **Confidence:** The rule holds in T with **confidence** $conf$ if $conf\%$ of transactions that contain X also contain Y .
 - $conf = \Pr(Y | X)$
- An association rule is a pattern that states when X occurs, Y occurs with certain probability.

Support and Confidence

- **Support count:** The support count of an itemset X , denoted by $X.count$, in a data set T is the number of transactions in T that contain X . Assume T has n transactions.
- Then,

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).
- **Key Features**
 - ❑ **Completeness:** find all rules.
 - ❑ **No target item(s)** on the right-hand-side
 - ❑ Mining with data on **hard disk** (not in memory)

An example



t1:	Beef, Chicken, Milk
t2:	Beef, Cheese
t3:	Cheese, Boots
t4:	Beef, Chicken, Cheese
t5:	Beef, Chicken, Clothes, Cheese, Milk
t6:	Chicken, Clothes, Milk
t7:	Chicken, Milk, Clothes

- Transaction data

- Assume:

minsup = 30%

minconf = 80%

- An example **frequent itemset**:

{Chicken, Clothes, Milk} [sup = 3/7]

- **Association rules** from the itemset:

Clothes → Milk, Chicken [sup = 3/7, conf = 3/3]

...

...

Clothes, Chicken → Milk, [sup = 3/7, conf = 3/3]

Transaction data representation

- A simplistic view of shopping baskets,
- Some important information not considered.
E.g,
 - the quantity of each item purchased and
 - the price paid.

Many mining algorithms

- There are a large number of them!!
- They use different strategies and data structures.
- Their resulting sets of rules are all the same.
 - Given a transaction data set T , and a minimum support and a minimum confident, the set of association rules existing in T is uniquely determined.
- Any algorithm should find the same set of rules although their computational efficiencies and memory requirements may be different.
- We study only one: the Apriori Algorithm

Road map

- Basic concepts of Association Rules
- **Apriori algorithm**
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

The Apriori algorithm

- The best known algorithm

- Two steps:

- Find all itemsets that have minimum support (*frequent itemsets*, also called large itemsets).
- Use frequent itemsets to **generate rules**.

- E.g., a frequent itemset

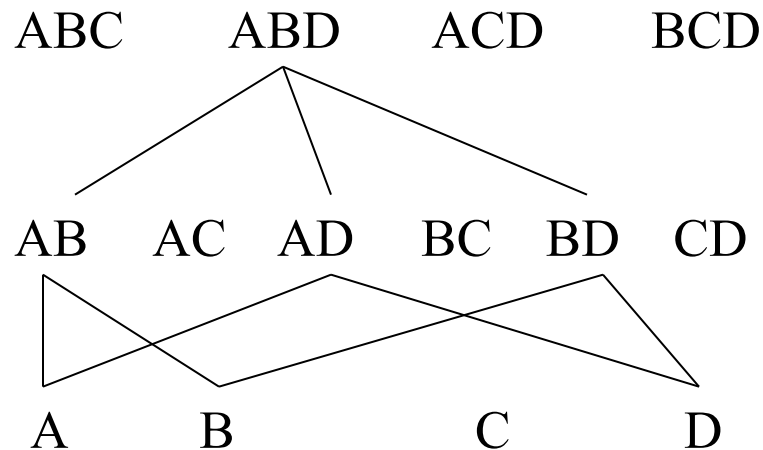
{Chicken, Clothes, Milk} [sup = 3/7]

and one rule from the frequent itemset

Clothes \rightarrow Milk, Chicken [sup = 3/7, conf = 3/3]

Step 1: Mining all frequent itemsets

- A **frequent *itemset*** is an itemset whose support is $\geq \text{minsup}$.
- **Key idea:** The **apriori property** (**downward closure property**): any subsets of a frequent itemset are also frequent itemsets



The Algorithm

- **Iterative algo.** (also called **level-wise search**):
Find all 1-item frequent itemsets; then all 2-item frequent itemsets, and so on.
 - In each iteration k , only consider itemsets that contain some $k-1$ frequent itemset.
- Find frequent itemsets of size 1: F_1
- **From $k = 2$**
 - C_k = candidates of size k : those itemsets of size k that could be frequent, given F_{k-1}
 - F_k = those itemsets that are actually frequent, $F_k \subseteq C_k$ (need to scan the database once).

Example – Finding frequent itemsets

Dataset T
minsup=0.5

TID	Items
T100	1, 3, 4
T200	2, 3, 5
T300	1, 2, 3, 5
T400	2, 5

itemset:count

1. scan T \rightarrow C_1 : {1}:2, {2}:3, {3}:3, {4}:1, {5}:3

\rightarrow F_1 : {1}:2, {2}:3, {3}:3, {5}:3

\rightarrow C_2 : {1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}

2. scan T \rightarrow C_2 : {1,2}:1, {1,3}:2, {1,5}:1, {2,3}:2, {2,5}:3, {3,5}:2

\rightarrow F_2 : {1,3}:2, {2,3}:2, {2,5}:3, {3,5}:2

\rightarrow C_3 : {2, 3,5}

3. scan T \rightarrow C_3 : {2, 3, 5}:2 \rightarrow F_3 : {2, 3, 5}

Details: ordering of items

- The items in I are sorted in **lexicographic order** (which is a total order).
- The order is used throughout the algorithm in each itemset.
- $\{w[1], w[2], \dots, w[k]\}$ represents a k -itemset w consisting of items $w[1], w[2], \dots, w[k]$, where $w[1] < w[2] < \dots < w[k]$ according to the total order.

Details: the algorithm

Algorithm Apriori(T)

```
 $C_1 \leftarrow \text{init-pass}(T);$   
 $F_1 \leftarrow \{f \mid f \in C_1, f.\text{count}/n \geq \text{minsup}\};$  //  $n$ : no. of transactions in  $T$   
for ( $k = 2$ ;  $F_{k-1} \neq \emptyset$ ;  $k++$ ) do  
     $C_k \leftarrow \text{candidate-gen}(F_{k-1});$   
    for each transaction  $t \in T$  do  
        for each candidate  $c \in C_k$  do  
            if  $c$  is contained in  $t$  then  
                 $c.\text{count}++$ ;  
            end  
        end  
     $F_k \leftarrow \{c \in C_k \mid c.\text{count}/n \geq \text{minsup}\}$   
end  
return  $F \leftarrow \bigcup_k F_k$ ;
```

Apriori candidate generation

- The **candidate-gen** function takes F_{k-1} and returns a **superset** (called the candidates) of the set of all **frequent k -itemsets**. It has two steps
 - **join step**: Generate all possible candidate itemsets C_k of length k
 - **prune step**: Remove those candidates in C_k that cannot be frequent.

Candidate-gen function

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

$C_k \leftarrow \emptyset$;

forall $f_1, f_2 \in F_{k-1}$

 with $f_1 = \{i_1, \dots, i_{k-2}, i_{k-1}\}$

 and $f_2 = \{i_1, \dots, i_{k-2}, i'_{k-1}\}$

 and $i_{k-1} < i'_{k-1}$ **do**

$c \leftarrow \{i_1, \dots, i_{k-1}, i'_{k-1}\}$; // join f_1 and f_2

$C_k \leftarrow C_k \cup \{c\}$;

for each $(k-1)$ -subset s of c **do**

if ($s \notin F_{k-1}$) **then**

 delete c from C_k ; // prune

end

end

return C_k ;

An example

- $F_3 = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{1, 3, 5\}, \{2, 3, 4\}\}$
- After join
 - $C_4 = \{\{1, 2, 3, 4\}, \{1, 3, 4, 5\}\}$
- After pruning:
 - $C_4 = \{\{1, 2, 3, 4\}\}$
because $\{1, 4, 5\}$ is not in F_3 ($\{1, 3, 4, 5\}$ is removed)

Step 2: Generating rules from frequent itemsets

- Frequent itemsets \neq association rules
- One more step is needed to generate association rules
- For each frequent itemset X ,
For each proper nonempty subset A of X ,
 - Let $B = X - A$
 - $A \rightarrow B$ is an association rule if
 - Confidence($A \rightarrow B$) \geq minconf,
support($A \rightarrow B$) = support($A \cup B$) = support(X)
confidence($A \rightarrow B$) = support($A \cup B$) / support(A)

Generating rules: an example

- Suppose $\{2,3,4\}$ is frequent, with $\text{sup}=50\%$
 - Proper nonempty subsets: $\{2,3\}$, $\{2,4\}$, $\{3,4\}$, $\{2\}$, $\{3\}$, $\{4\}$, with $\text{sup}=50\%$, 50% , 75% , 75% , 75% , 75% respectively
 - These generate these association rules:
 - $2,3 \rightarrow 4$, confidence= 100%
 - $2,4 \rightarrow 3$, confidence= 100%
 - $3,4 \rightarrow 2$, confidence= 67%
 - $2 \rightarrow 3,4$, confidence= 67%
 - $3 \rightarrow 2,4$, confidence= 67%
 - $4 \rightarrow 2,3$, confidence= 67%
 - All rules have support = 50%

Generating rules: summary

- To recap, in order to obtain $A \rightarrow B$, we need to have $\text{support}(A \cup B)$ and $\text{support}(A)$
- All the required information for confidence computation has already been recorded in itemset generation. No need to see the data T any more.
- This step is not as time-consuming as frequent itemsets generation.

On Apriori Algorithm

Seems to be very expensive

- Level-wise search
- K = the size of the largest itemset
- It makes at most K passes over data
- In practice, K is bounded (10).
- The algorithm is very fast. Under some conditions, all rules can be found in **linear time**.
- Scale up to large data sets

More on association rule mining

- Clearly the space of all association rules is **exponential, $O(2^m)$** , where m is the number of items in I .
- The mining exploits **sparseness of data**, and **high minimum support** and **high minimum confidence** values.
- Still, it always produces a **huge number of rules**, thousands, tens of thousands, millions, ...

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

Different data formats for mining

- The data can be in transaction form or table form

Transaction form:

a, b
a, c, d, e
a, d, f

Table form:

Attr1	Attr2	Attr3
a,	b,	d
b,	c,	e

- Table data need to be converted to transaction form for association mining

From a table to a set of transactions

Table form:

Attr1	Attr2	Attr3
a,	b,	d
b,	c,	e

⇒ Transaction form:

(Attr1, a), (Attr2, b), (Attr3, d)

(Attr1, b), (Attr2, c), (Attr3, e)

candidate-gen can be slightly improved. Why?

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

Problems with the association mining

- **Single minsup:** It assumes that all items in the data are of the **same nature** and/or have **similar frequencies**.
- **Not true:** In many applications, some items appear very frequently in the data, while others rarely appear.
E.g., in a supermarket, people buy *food processor* and *cooking pan* much less frequently than they buy *bread* and *milk*.

Rare Item Problem

- If the frequencies of items vary a great deal, we will encounter **two problems**
 - ❑ If **minsup** is set too high, those rules that involve rare items will not be found.
 - ❑ To find rules that involve both frequent and rare items, **minsup has to be set very low**. This may cause **combinatorial explosion** because those frequent items will be associated with one another in all possible ways.

Multiple minsups model

- The minimum support of a rule is expressed in terms of *minimum item supports (MIS)* of the items that appear in the rule.
- Each item can have a *minimum item support*.
- By providing different MIS values for different items, the user effectively expresses different support requirements for different rules.
- To prevent very frequent items and very rare items from appearing in the same itemsets, we introduce a **support difference constraint**.

$$\max_{i \in S} \{sup\{i\}\} - \min_{i \in S} \{sup(i)\} \leq \varphi,$$

Minsup of a rule

- Let $MIS(i)$ be the MIS value of item i . The *minsup* of a rule R is the lowest MIS value of the items in the rule.
- I.e., a rule R : $a_1, a_2, \dots, a_k \rightarrow a_{k+1}, \dots, a_r$ satisfies its minimum support if its actual support is \geq
 $\min(MIS(a_1), MIS(a_2), \dots, MIS(a_r)).$

An Example

- Consider the following items:

bread, shoes, clothes

The user-specified MIS values are as follows:

$MIS(bread) = 2\%$ $MIS(shoes) = 0.1\%$

$MIS(clothes) = 0.2\%$

The following rule **doesn't satisfy its minsup**:

clothes \rightarrow *bread* [sup=0.15%, conf =70%]

The following rule **satisfies its minsup**:

clothes \rightarrow *shoes* [sup=0.15%, conf =70%]

Downward closure property

- In the new model, **the property no longer holds (?)**

E.g., Consider four items 1, 2, 3 and 4 in a database. Their minimum item supports are

$$\text{MIS}(1) = 10\%$$

$$\text{MIS}(2) = 20\%$$

$$\text{MIS}(3) = 5\%$$

$$\text{MIS}(4) = 6\%$$

$\{1, 2\}$ with support 9% is infrequent, but $\{1, 2, 3\}$ and $\{1, 2, 4\}$ could be frequent.

To deal with the problem

- We sort all items in I according to their MIS values (make it a total order).
- The order is used throughout the algorithm in each itemset.
- Each itemset w is of the following form:
 $\{w[1], w[2], \dots, w[k]\}$, consisting of items,
 $w[1], w[2], \dots, w[k]$,
where $\text{MIS}(w[1]) \leq \text{MIS}(w[2]) \leq \dots \leq \text{MIS}(w[k])$.

The MSapriori algorithm

Algorithm MSapriori(T, MS, φ) // φ is for support difference constraint

```
 $M \leftarrow \text{sort}(I, MS);$   
 $L \leftarrow \text{init-pass}(M, T);$   
 $F_1 \leftarrow \{\{i\} \mid i \in L, i.\text{count}/n \geq \text{MIS}(i)\};$   
for ( $k = 2; F_{k-1} \neq \emptyset; k++$ ) do  
    if  $k=2$  then  
         $C_k \leftarrow \text{level2-candidate-gen}(L, \varphi)$   
    else  $C_k \leftarrow \text{MSCandidate-gen}(F_{k-1}, \varphi);$   
    end;  
    for each transaction  $t \in T$  do  
        for each candidate  $c \in C_k$  do  
            if  $c$  is contained in  $t$  then  
                 $c.\text{count}++;$   
                if  $c - \{c[1]\}$  is contained in  $t$  then  
                     $c.\text{tailCount}++$   
            end  
        end  
         $F_k \leftarrow \{c \in C_k \mid c.\text{count}/n \geq \text{MIS}(c[1])\}$   
    end  
return  $F \leftarrow \bigcup_k F_k;$ 
```


Candidate itemset generation

- **Special treatments needed:**
 - Sorting the items according to their MIS values
 - First pass over data (the first three lines)
 - Let us look at this in detail.
 - Candidate generation at level-2
 - Read it in the handout.
 - Pruning step in level- k ($k > 2$) candidate generation.
 - Read it in the handout.

First pass over data

- It makes a pass over the data to record the support count of each item.
- It then follows the sorted order to find the first item i in M that meets $\text{MIS}(i)$.
 - i is inserted into L .
 - For each subsequent item j in M after i , if $j.\text{count}/n \geq \text{MIS}(i)$ then j is also inserted into L , where $j.\text{count}$ is the support count of j and n is the total number of transactions in T . Why?
- L is used by function level2-candidate-gen

First pass over data: an example

- Consider the four items 1, 2, 3 and 4 in a data set. Their minimum item supports are:
 $MIS(1) = 10\%$ $MIS(2) = 20\%$
 $MIS(3) = 5\%$ $MIS(4) = 6\%$
- Assume our data set has 100 transactions. The first pass gives us the following support counts:
 $\{3\}.count = 6$, $\{4\}.count = 3$,
 $\{1\}.count = 9$, $\{2\}.count = 25$.
- **Then $L = \{3, 1, 2\}$, and $F_1 = \{\{3\}, \{2\}\}$**
- Item 4 is not in L because $4.count/n < MIS(3)$ ($= 5\%$),
- $\{1\}$ is not in F_1 because $1.count/n < MIS(1)$ ($= 10\%$).

Rule generation

- The following two lines in MSapriori algorithm are important for rule generation, which are not needed for the Apriori algorithm
if $c - \{c[1]\}$ is contained in t **then**
 c.tailCount++
- Many rules cannot be generated without them.
- Why?

On multiple minsup rule mining

- Multiple minsup model **subsumes** the single support model.
- It is a **more realistic** model for practical applications.
- The model enables us to found **rare item rules** yet without producing a huge number of meaningless rules with frequent items.
- By setting MIS values of some items to 100% (or more), we effectively instruct the algorithms not to generate rules only involving these items.

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

Mining class association rules (CAR)

- Normal association rule mining does not have any target.
- It finds all possible rules that exist in data, i.e., any item can appear as a consequent or a condition of a rule.
- However, in some applications, the user is interested in some targets.
 - E.g, the user has a set of text documents from some known topics. He/she wants to find out what words are associated or correlated with each topic.

Problem definition

- Let T be a transaction data set consisting of n transactions.
- Each transaction is also labeled with a class y .
- Let I be the set of all items in T , Y be the set of all class labels and $I \cap Y = \emptyset$.
- A **class association rule (CAR)** is an implication of the form
$$X \rightarrow y, \text{ where } X \subseteq I, \text{ and } y \in Y.$$
- The definitions of **support** and **confidence** are the same as those for normal association rules.

An example

- **A text document data set**

doc 1:	Student, Teach, School	: Education
doc 2:	Student, School	: Education
doc 3:	Teach, School, City, Game	: Education
doc 4:	Baseball, Basketball	: Sport
doc 5:	Basketball, Player, Spectator	: Sport
doc 6:	Baseball, Coach, Game, Team	: Sport
doc 7:	Basketball, Team, City, Game	: Sport

- Let $minsup = 20\%$ and $minconf = 60\%$. The following are two examples of class association rules:

Student, School \rightarrow Education	[sup= 2/7, conf = 2/2]
game \rightarrow Sport	[sup= 2/7, conf = 2/3]

Mining algorithm

- Unlike normal association rules, CARs can be mined directly in one step.
- The key operation is to find all **ruleitems** that have support above *minsup*. A **ruleitem** is of the form:

$(condset, y)$

where **condset** is a set of items from I (i.e., $condset \subseteq I$), and $y \in Y$ is a class label.

- Each ruleitem basically represents a rule:

$condset \rightarrow y,$

- The Apriori algorithm can be modified to generate CARs

Multiple minimum class supports

- The multiple minimum support idea can also be applied here.
- The user can specify different **minimum supports to different classes**, which effectively assign a different minimum support to rules of each class.
- For example, we have a data set with two classes, Yes and No. We may want
 - rules of class Yes to have the minimum support of 5% and
 - rules of class No to have the minimum support of 10%.
- By setting minimum class supports to 100% (or more for some classes), we tell the algorithm not to generate rules of those classes.
 - This is a very useful trick in applications.

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

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

Basic concepts

- Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of items.
- **Sequence**: An ordered list of itemsets.
- **Itemset/element**: A non-empty set of items $X \subseteq I$.
We denote a sequence s by $\langle a_1 a_2 \dots a_r \rangle$, where a_i is an itemset, which is also called an **element** of s .
- An element (or an itemset) of a sequence is denoted by $\{x_1, x_2, \dots, x_k\}$, where $x_j \in I$ is an item.
- We assume without loss of generality that items in an element of a sequence are in **lexicographic order**.

Basic concepts (contd)

- **Size**: The **size** of a sequence is the number of elements (or itemsets) in the sequence.
- **Length**: The **length** of a sequence is the number of items in the sequence.
 - A sequence of length k is called **k -sequence**.
- A sequence $s_1 = \langle a_1 a_2 \dots a_r \rangle$ is a **subsequence** of another sequence $s_2 = \langle b_1 b_2 \dots b_v \rangle$, or s_2 is a **supersequence** of s_1 , if there exist integers $1 \leq j_1 < j_2 < \dots < j_{r-1} < j_r \leq v$ such that $a_1 \subseteq b_{j_1}$, $a_2 \subseteq b_{j_2}$, ..., $a_r \subseteq b_{j_r}$. We also say that s_2 **contains** s_1 .

An example

- 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.

Objective

- Given a set S of **input data sequences** (or sequence database), the problem of mining sequential patterns is to find all the sequences that have **a user-specified minimum support**.
- Each such sequence is called a **frequent sequence**, or a **sequential pattern**.
- The **support** for a sequence is the fraction of total data sequences in S that contains this sequence.

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	$\langle \{30\} \{90\} \rangle$
2	$\langle \{10, 20\} \{30\} \{40, 60, 70\} \rangle$
3	$\langle \{30, 50, 70\} \rangle$
4	$\langle \{30\} \{40, 70\} \{90\} \rangle$
5	$\langle \{90\} \rangle$

Table 3. The final output sequential patterns

	Sequential Patterns with Support $\geq 25\%$
1-sequences	$\langle \{30\} \rangle, \langle \{40\} \rangle, \langle \{70\} \rangle, \langle \{90\} \rangle$
2-sequences	$\langle \{30\} \{40\} \rangle, \langle \{30\} \{70\} \rangle, \langle \{30\} \{90\} \rangle, \langle \{40, 70\} \rangle$
3-sequences	$\langle \{30\} \{40, 70\} \rangle$

GSP mining algorithm

- Very similar to the Apriori algorithm

Algorithm GSP(S)

```
1   $C_1 \leftarrow \text{init-pass}(S);$  // the first pass over  $S$ 
2   $F_1 \leftarrow \{\langle \{f\} \rangle \mid f \in C_1, f.\text{count}/n \geq \text{minsup}\};$  //  $n$  is the number of sequences in  $S$ 
3  for ( $k = 2; F_{k-1} \neq \emptyset; k++$ ) do // subsequent passes over  $S$ 
4     $C_k \leftarrow \text{candidate-gen-SPM}(F_{k-1});$ 
5    for each data sequence  $s \in S$  do // scan the data once
6      for each candidate  $c \in C_k$  do
7        if  $c$  is contained in  $s$  then
8           $c.\text{count}++;$  // increment the support count
9        end
10   end
11    $F_k \leftarrow \{c \in C_k \mid c.\text{count}/n \geq \text{minsup}\}$ 
12 end
13 return  $\bigcup_k F_k;$ 
```

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
$\langle \{1, 2\} \{4\} \rangle$	$\langle \{1, 2\} \{4, 5\} \rangle$	$\langle \{1, 2\} \{4, 5\} \rangle$
$\langle \{1, 2\} \{5\} \rangle$	$\langle \{1, 2\} \{4\} \{6\} \rangle$	
$\langle \{1\} \{4, 5\} \rangle$		
$\langle \{1, 4\} \{6\} \rangle$		
$\langle \{2\} \{4, 5\} \rangle$		
$\langle \{2\} \{4\} \{6\} \rangle$		

Now it is your turn ...

Programming assignment!

- Implement two algorithms for sequential pattern mining considering
 - multiple minimum supports
 - support difference constraint
- **Algorithms:** (1) MS-GSP, and (2) MSprefixSpan
- Each group implements only 1 algorithm
 - **Deadline: Sept 27, 2006** (Demo your program on that day)
 - Test data sequences will be in one file in the same format as those in the book.

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining with multiple minimum supports
- Mining class association rules
- Sequential pattern mining
- Summary

Summary

- Association rule mining has been extensively studied in the data mining community.
- So is sequential pattern mining
- There are many efficient algorithms and model variations.
- Other related work includes
 - Multi-level or generalized rule mining
 - Constrained rule mining
 - Incremental rule mining
 - Maximal frequent itemset mining
 - Closed itemset mining
 - Rule interestingness and visualization
 - Parallel algorithms
 - ...