

Data Mining

資料探勘

關連分析

(Association Analysis)

1002DM02

MI4

Thu. 9,10 (16:10-18:00) B513

Min-Yuh Day

戴敏育

Assistant Professor

專任助理教授

Dept. of Information Management, Tamkang University

淡江大學 資訊管理學系

<http://mail.tku.edu.tw/myday/>

2012-02-23

課程大綱 (Syllabus)

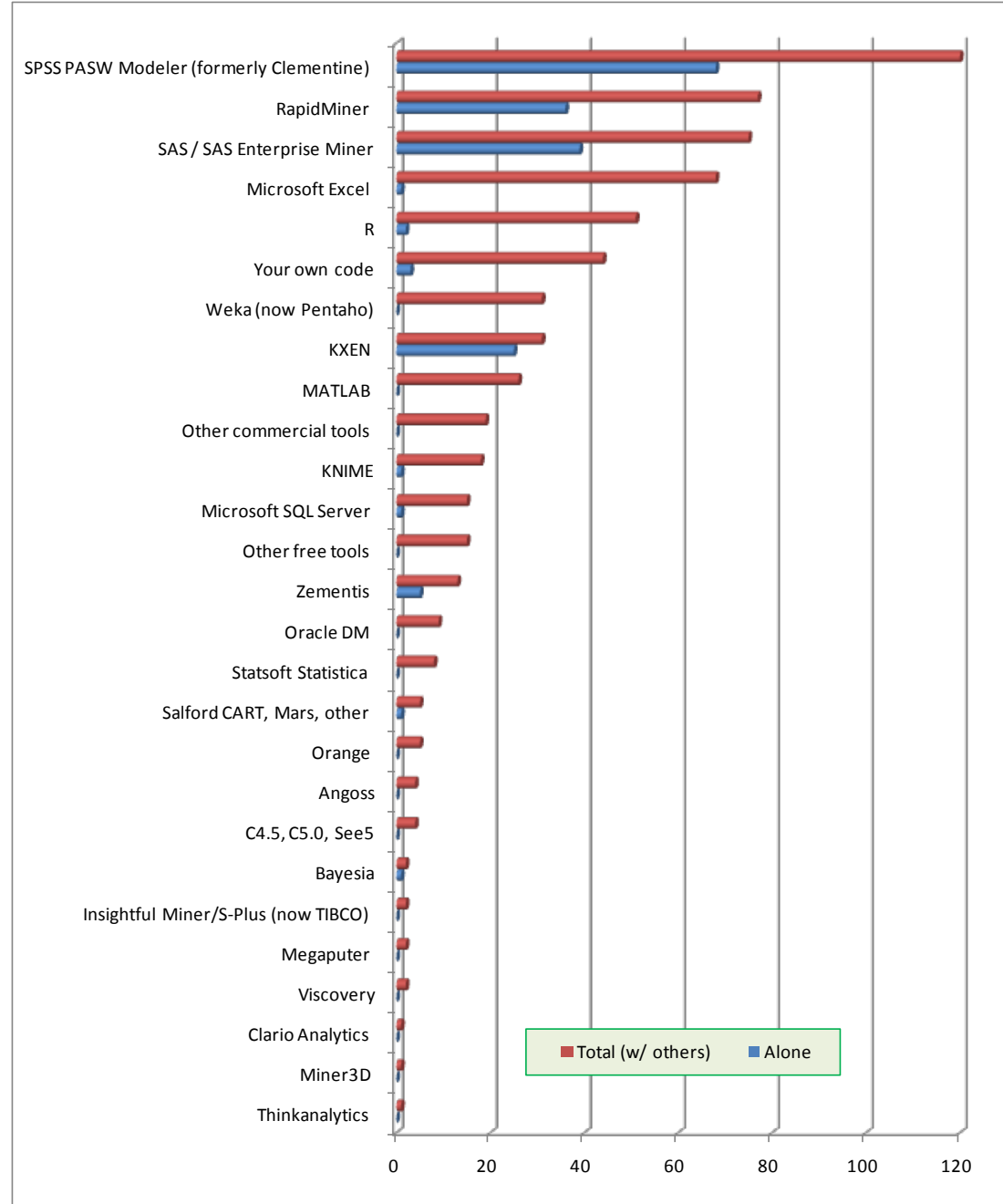
週次	日期	內容 (Subject/Topics)
1	101/02/16	資料探勘導論 (Introduction to Data Mining)
2	101/02/23	關連分析 (Association Analysis)
3	101/03/01	分類與預測 (Classification and Prediction)
4	101/03/08	分群分析 (Cluster Analysis)
5	101/03/15	個案分析與實作一 (分群分析)： Banking Segmentation (Cluster Analysis – KMeans)
6	101/03/22	個案分析與實作二 (關連分析)： Web Site Usage Associations (Association Analysis)
7	101/03/29	個案分析與實作三 (決策樹、模型評估)： Enrollment Management Case Study (Decision Tree, Model Evaluation)

課程大綱 (Syllabus)

週次	日期	內容 (Subject/Topics)
8	101/04/05	教學行政觀摩日 (--No Class--)
9	101/04/12	期中報告 (Midterm Presentation)
10	101/04/19	期中考試週
11	101/04/26	個案分析與實作四 (迴歸分析、類神經網路) : Credit Risk Case Study (Regression Analysis, Artificial Neural Network)
12	101/05/03	文字探勘與網頁探勘 (Text and Web Mining)
13	101/05/10	社會網路分析、意見分析 (Social Network Analysis, Opinion Mining)
14	101/05/17	期末專題報告 (Term Project Presentation)
15	101/05/24	畢業考試週

Data Mining Software

- Commercial
 - SPSS - PASW (formerly Clementine)
 - SAS - Enterprise Miner
 - IBM - Intelligent Miner
 - StatSoft – Statistical Data Miner
 - ... many more
- Free and/or Open Source
 - Weka
 - RapidMiner...



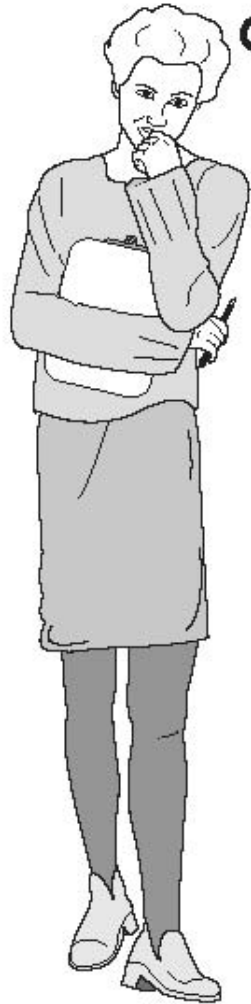
Source: KDNuggets.com, May 2009

Association Analysis: Mining Frequent Patterns, Association and Correlations

- Association Analysis
- Mining Frequent Patterns
- Association and Correlations
- Apriori Algorithm

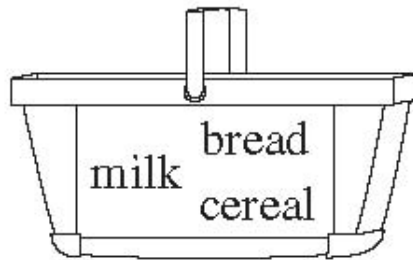
Market Basket Analysis

Which items are frequently purchased together by my customers?

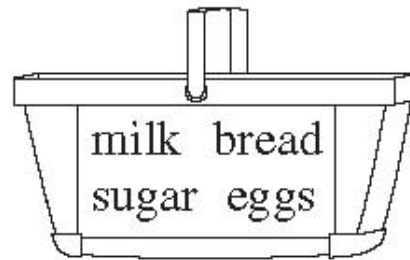


Market Analyst

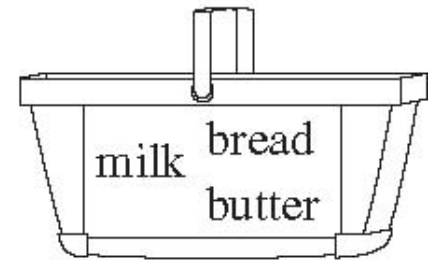
Shopping Baskets



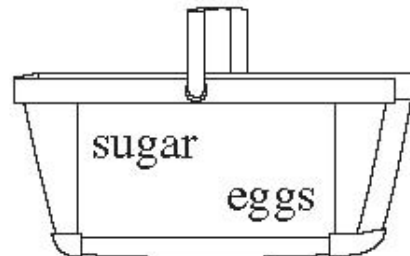
Customer 1



Customer 2



Customer 3



Customer n

Association Rule Mining

- Apriori Algorithm

Raw Transaction Data

Transaction No	SKUs (Item No)
1	1, 2, 3, 4
1	2, 3, 4
1	2, 3
1	1, 2, 4
1	1, 2, 3, 4
1	2, 4

One-item Itemsets

Itemset (SKUs)	Support
1	3
2	6
3	4
4	5

Two-item Itemsets

Itemset (SKUs)	Support
1, 2	3
1, 3	2
1, 4	3
2, 3	4
2, 4	5
3, 4	3

Three-item Itemsets

Itemset (SKUs)	Support
1, 2, 4	3
2, 3, 4	3

Association Rule Mining

- A very popular DM method in business
- Finds interesting relationships (affinities) between variables (items or events)
- Part of machine learning family
- Employs unsupervised learning
- There is no output variable
- Also known as **market basket analysis**
- Often used as an example to describe DM to ordinary people, such as the famous “relationship between diapers and beers!”

Association Rule Mining

- **Input:** the simple point-of-sale transaction data
- **Output:** Most frequent affinities among items
- Example: according to the transaction data...

“Customer who bought a laptop computer and a virus protection software, also bought extended service plan 70 percent of the time.”
- How do you use such a pattern/knowledge?
 - Put the items next to each other for ease of finding
 - Promote the items as a package (do not put one on sale if the other(s) are on sale)
 - Place items far apart from each other so that the customer has to walk the aisles to search for it, and by doing so potentially seeing and buying other items

Association Rule Mining

- A representative applications of association rule mining include
 - **In business:** cross-marketing, cross-selling, store design, catalog design, e-commerce site design, optimization of online advertising, product pricing, and sales/promotion configuration
 - **In medicine:** relationships between symptoms and illnesses; diagnosis and patient characteristics and treatments (to be used in medical DSS); and genes and their functions (to be used in genomics projects)...

Association Rule Mining

- Are all association rules interesting and useful?

A Generic Rule: $X \Rightarrow Y$ [S%, C%]

X, Y: products and/or services

X: Left-hand-side (LHS)

Y: Right-hand-side (RHS)

S: **Support:** how often **X** and **Y** go together

C: **Confidence:** how often **Y** go together with the **X**

Example: {Laptop Computer, Antivirus Software} \Rightarrow
{Extended Service Plan} [30%, 70%]

Association Rule Mining

- Algorithms are available for generating association rules
 - Apriori
 - Eclat
 - FP-Growth
 - + Derivatives and hybrids of the three
- The algorithms help identify the **frequent item sets**, which are, then converted to association rules

Association Rule Mining

- Apriori Algorithm
 - Finds subsets that are common to at least a minimum number of the itemsets
 - uses a bottom-up approach
 - frequent subsets are extended one item at a time (the size of frequent subsets increases from one-item subsets to two-item subsets, then three-item subsets, and so on), and
 - groups of candidates at each level are tested against the data for minimum

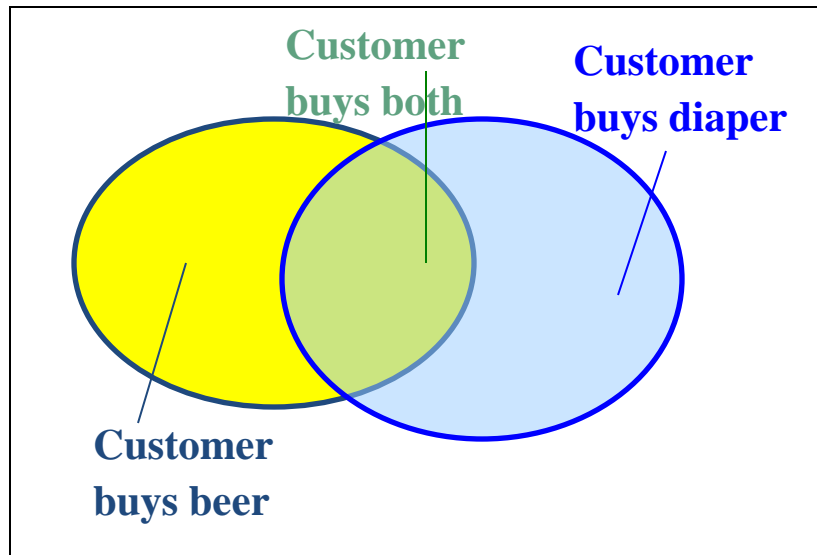
What Is Frequent Pattern Analysis?

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
 - What products were often purchased together?
 - Beer and diapers?!
 - What are the subsequent purchases after buying a PC?
 - What kinds of DNA are sensitive to this new drug?
 - Can we automatically classify web documents?
- Applications
 - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

- Itemset $X = \{x_1, \dots, x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
 - **support**, s , **probability** that a transaction contains $X \cup Y$
 - **confidence**, c , **conditional probability** that a transaction having X also contains Y



Let $sup_{min} = 50\%$, $conf_{min} = 50\%$
 Freq. Pat.: $\{A:3, B:3, D:4, E:3, AD:3\}$

Association rules:

$A \rightarrow D$ (60%, 100%)

$D \rightarrow A$ (60%, 75%)

$A \rightarrow D$ (support = $3/5 = 60\%$, confidence = $3/3 = 100\%$)

$D \rightarrow A$ (support = $3/5 = 60\%$, confidence = $3/4 = 75\%$)

Market basket analysis

- Example
 - Which groups or sets of items are customers likely to purchase on a given trip to the store?
- Association Rule
 - *Computer* → *antivirus_software*
[support = 2%; confidence = 60%]
 - A support of 2% means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
 - A confidence of 60% means that 60% of the customers who purchased a computer also bought the software.

Association rules

- Association rules are considered interesting if they satisfy both
 - a **minimum support threshold** and
 - a **minimum confidence threshold**.

Frequent Itemsets, Closed Itemsets, and Association Rules

Let $I = \{I_1, I_2, \dots, I_m\}$ be a set of items. Let D , the task-relevant data, be a set of database transactions where each transaction T is a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B = \phi$. The rule $A \Rightarrow B$ holds in the transaction set D with support s , where s is the percentage of transactions in D that contain $A \cup B$ (i.e., the union of sets A and B , or say, both A and B). This is taken to be the probability, $P(A \cup B)$.¹ The rule $A \Rightarrow B$ has confidence c in the transaction set D , where c is the percentage of transactions in D containing A that also contain B . This is taken to be the conditional probability, $P(B|A)$. That is,

$$\text{Support } (A \rightarrow B) = P(A \cup B)$$

$$\text{Confidence } (A \rightarrow B) = P(B|A)$$

Support $(A \rightarrow B) = P(A \cup B)$
Confidence $(A \rightarrow B) = P(B|A)$

- The notation $P(A \cup B)$ indicates the probability that a transaction contains the union of set A and set B
 - (i.e., it contains every item in A and in B).
- This should not be confused with $P(A \text{ or } B)$, which indicates the probability that a transaction contains either A or B .

- Rules that satisfy both a **minimum support threshold (*min_sup*)** and a **minimum confidence threshold (*min_conf*)** are called **strong**.
- By convention, we write support and confidence values so as to occur between 0% and 100%, rather than 0 to 1.0.

- itemset
 - A set of items is referred to as an **itemset**.
- K-itemset
 - An itemset that contains *k items* is a **k-itemset**.
- Example:
 - The set {*computer, antivirus software*} is a **2-itemset**.

Absolute Support and Relative Support

- Absolute Support

- The **occurrence frequency** of an itemset is the number of transactions that contain the itemset
 - frequency, support count, or count of the itemset
- Ex: 3

- Relative support

- Ex: 60%

- If the **relative support** of an itemset I satisfies a **prespecified minimum support threshold**, then I is a **frequent itemset**.
 - i.e., the **absolute support** of I satisfies the corresponding **minimum support count threshold**
- The set of **frequent k -itemsets** is commonly denoted by L_K

$$\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\text{support}(A \cup B)}{\text{support}(A)} = \frac{\text{support_count}(A \cup B)}{\text{support_count}(A)}$$

- the **confidence** of rule $A \rightarrow B$ can be easily derived from the support counts of A and $A \cup B$.
- once the support counts of A , B , and $A \cup B$ are found, it is straightforward to derive the corresponding association rules $A \rightarrow B$ and $B \rightarrow A$ and check whether they are strong.
- Thus the problem of mining association rules can be reduced to that of mining frequent itemsets.

Association rule mining: Two-step process

1. Find all frequent itemsets

- By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min_sup*.

2. Generate strong association rules from the frequent itemsets

- By definition, these rules must satisfy minimum support and minimum confidence.

Closed frequent itemsets and maximal frequent itemsets

- Suppose that a transaction database has only two transactions:
 - $\{(a_1, a_2, \dots, a_{100}); (a_1, a_2, \dots, a_{50})\}$
- Let the minimum support count threshold be $min_sup=1$.
- We find two **closed frequent itemsets** and their support counts, that is,
 - $C = \{\{a_1, a_2, \dots, a_{100}\}:1; \{a_1, a_2, \dots, a_{50}\}: 2\}$
- There is one **maximal frequent itemset**:
 - $M = \{\{a_1, a_2, \dots, a_{100}\}:1\}$
 - (We cannot include $\{a_1, a_2, \dots, a_{50}\}$ as a maximal frequent itemset because it has a frequent super-set, $\{a_1, a_2, \dots, a_{100}\}$)

Frequent Pattern Mining

- Based on the *completeness of patterns to be mined*
- Based on the *levels of abstraction involved in the rule set*
- Based on the *number of data dimensions involved in the rule*
- Based on the *types of values handled in the rule*
- Based on the *kinds of rules to be mined*
- Based on the *kinds of patterns to be mined*

Based on the *levels of abstraction* *involved in the rule set*

- $buys(X, \text{"computer"}) \rightarrow buys(X, \text{"HP printer"})$
- $buys(X, \text{"laptop computer"}) \rightarrow buys(X, \text{"HP printer"})$

Based on the *number of data dimensions involved in the rule*

- Single-dimensional association rule
 - $\text{buys}(X, \text{"computer"}) \rightarrow \text{buys}(X, \text{"antivirus software"})$
- Multidimensional association rule
 - $\text{age}(X, \text{"30,...,39"}) \wedge \text{income}(X, \text{"42K,...,48K"}) \rightarrow \text{buys}(X, \text{"high resolution TV"})$

Efficient and Scalable Frequent Itemset Mining Methods

- The Apriori Algorithm
 - Finding Frequent Itemsets Using Candidate Generation

Apriori Algorithm

- **Apriori** is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules.
- The name of the algorithm is based on the fact that the algorithm uses *prior knowledge of frequent itemset properties*, as we shall see following.

Apriori Algorithm

- Apriori employs an iterative approach known as a *level-wise search*, where *k*-itemsets are used to explore *(k+1)*-itemsets.
- First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted L_1 .
- Next, L_1 is used to find L_2 , the set of frequent 2-itemsets, which is used to find L_3 , and so on, until no more frequent *k*-itemsets can be found.
- The finding of each L_k requires one full scan of the database.

Apriori Algorithm

- To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the **Apriori property**.
- Apriori property
 - *All nonempty subsets of a frequent itemset must also be frequent.*

- *How is the Apriori property used in the algorithm?*
 - How L_{k-1} is used to find L_k for $k \geq 2$.
 - A two-step process is followed, consisting of **join** and **prune** actions.

Apriori property used in algorithm

1. The join step

1. **The join step:** To find L_k , a set of candidate k -itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k . Let l_1 and l_2 be itemsets in L_{k-1} . The notation $l_i[j]$ refers to the j th item in l_i (e.g., $l_1[k-2]$ refers to the second to the last item in l_1). By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. For the $(k-1)$ -itemset, l_i , this means that the items are sorted such that $l_i[1] < l_i[2] < \dots < l_i[k-1]$. The join, $L_{k-1} \bowtie L_{k-1}$, is performed, where members of L_{k-1} are joinable if their first $(k-2)$ items are in common. That is, members l_1 and l_2 of L_{k-1} are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \dots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. The condition $l_1[k-1] < l_2[k-1]$ simply ensures that no duplicates are generated. The resulting itemset formed by joining l_1 and l_2 is $l_1[1], l_1[2], \dots, l_1[k-2], l_1[k-1], l_2[k-1]$.

Apriori property used in algorithm

2. The prune step

2. The prune step: C_k is a superset of L_k , that is, its members may or may not be frequent, but all of the frequent k -itemsets are included in C_k . A scan of the database to determine the count of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k). C_k , however, can be huge, and so this could involve heavy computation. To reduce the size of C_k , the Apriori property is used as follows. Any $(k - 1)$ -itemset that is not frequent cannot be a subset of a frequent k -itemset. Hence, if any $(k - 1)$ -subset of a candidate k -itemset is not in L_{k-1} , then the candidate cannot be frequent either and so can be removed from C_k . This subset testing can be done quickly by maintaining a hash tree of all frequent itemsets.

Transactional data for an *AllElectronics* branch

<i>TID</i>	<i>List of item IDs</i>
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Example: Apriori

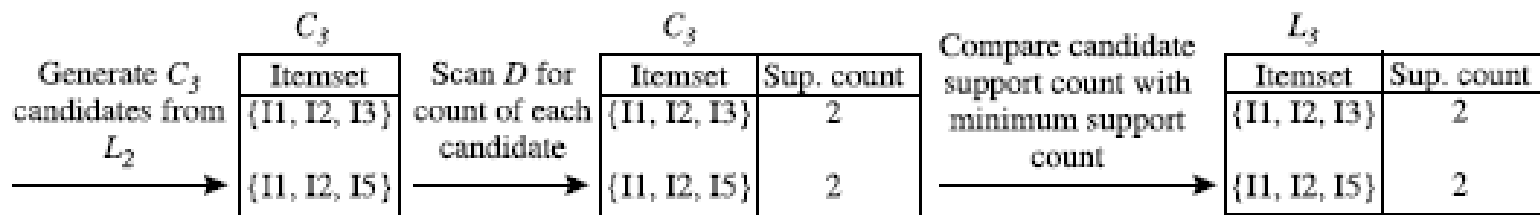
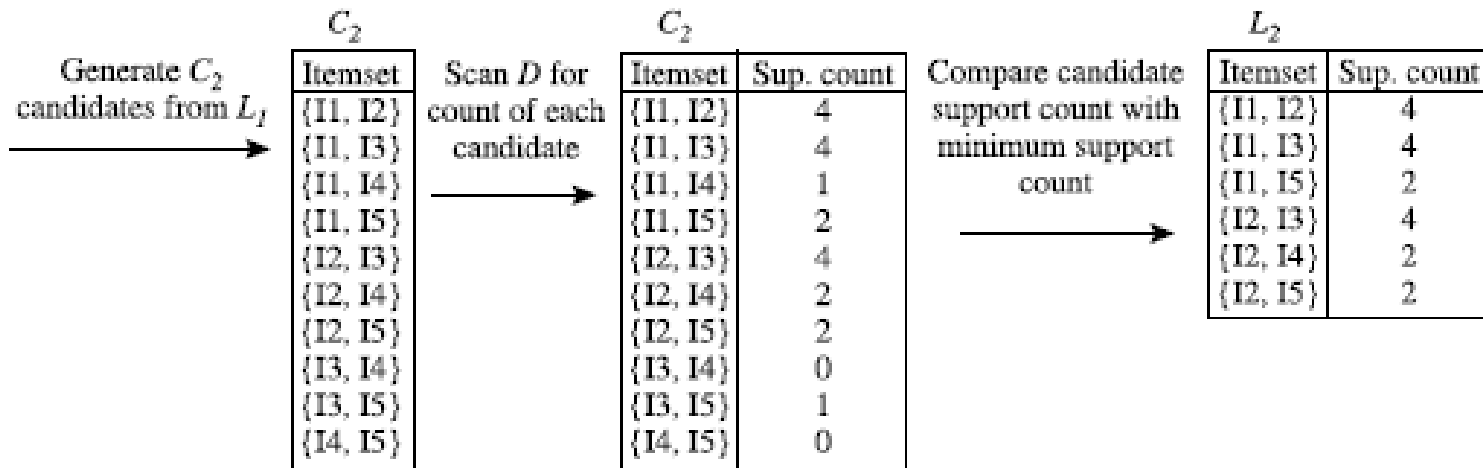
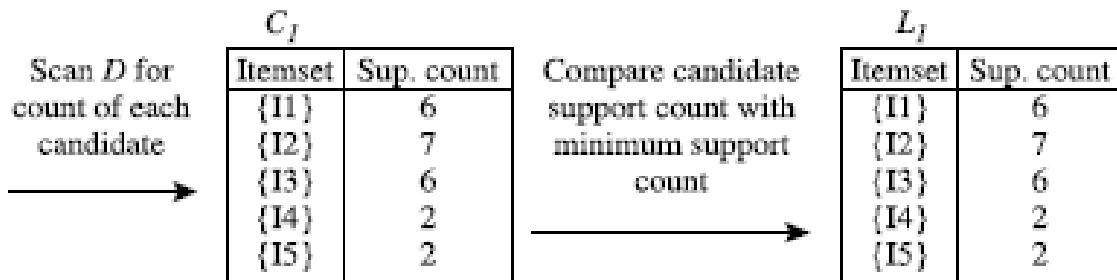
- Let's look at a concrete example, based on the *AllElectronics transaction database, D*.
- *There are nine transactions in this database, that is, $|D| = 9$.*
- Apriori algorithm for finding frequent itemsets in *D*

<i>TID</i>	<i>List of item_IDs</i>
T100	11, 12, 15
T200	12, 14
T300	12, 13
T400	11, 12, 14
T500	11, 13
T600	12, 13
T700	11, 13
T800	11, 12, 13, 15
T900	11, 12, 13

Example: Apriori Algorithm

Generation of candidate itemsets and frequent itemsets,
where the minimum support count is 2.

TID	List of item_IDs
T100	11, 12, 15
T200	12, 14
T300	12, 13
T400	11, 12, 14
T500	11, 13
T600	12, 13
T700	11, 13
T800	11, 12, 13, 15
T900	11, 12, 13



Example: Apriori Algorithm

$$C_1 \rightarrow L_1$$

<i>TID</i>	<i>List of item_IDs</i>
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Scan *D* for
count of each
candidate



C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

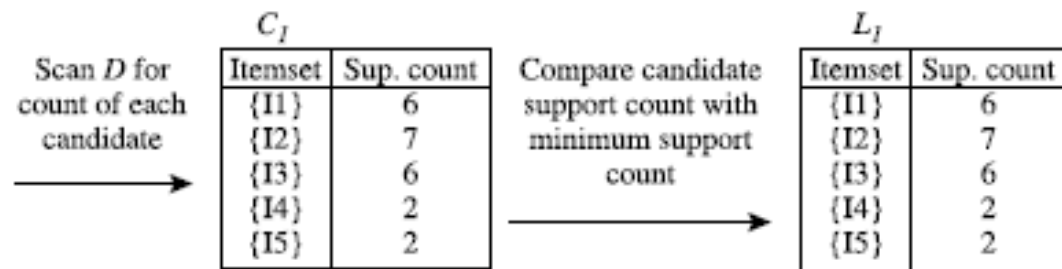
Compare candidate
support count with
minimum support
count



L_1

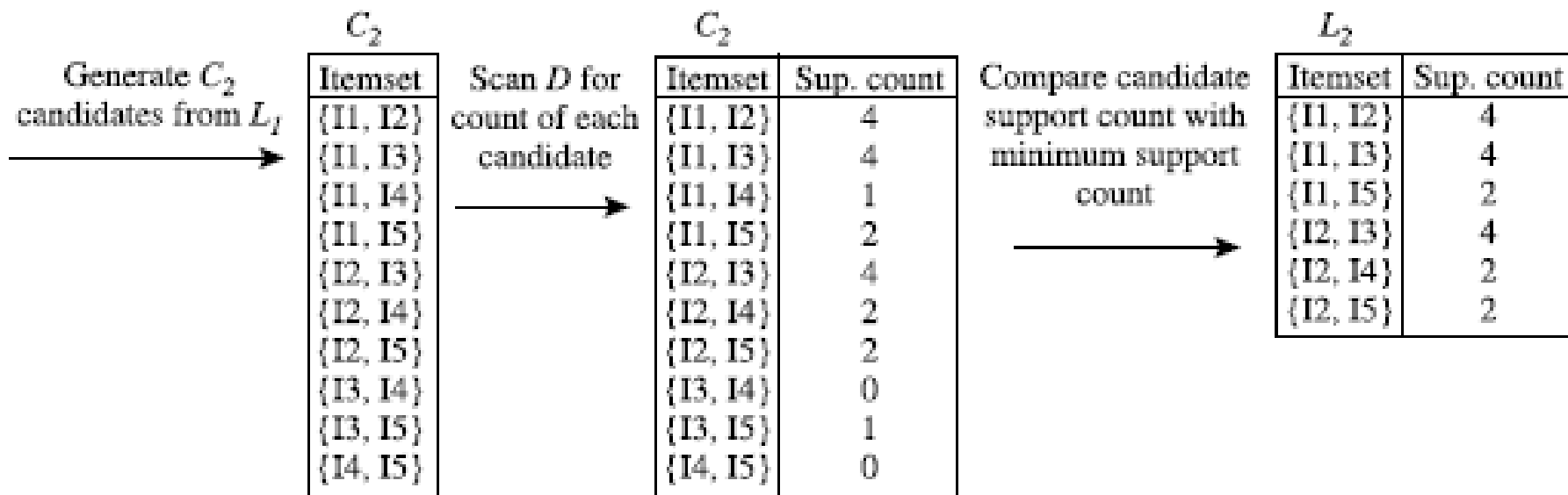
Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

<i>TID</i>	<i>List of item_IDs</i>
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

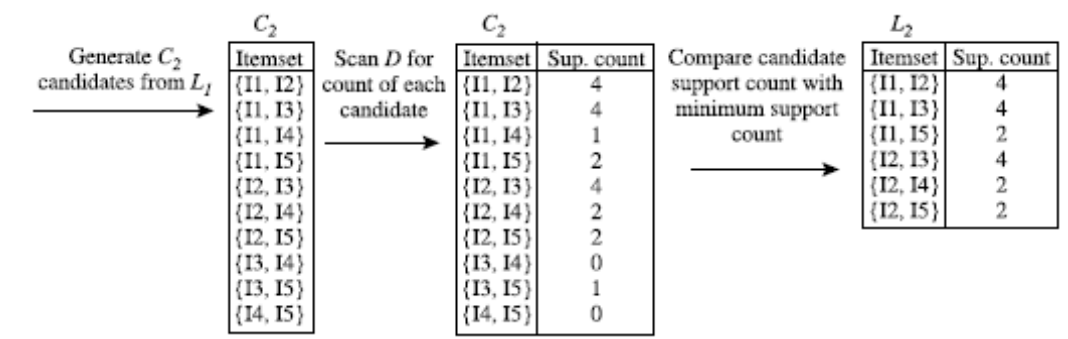
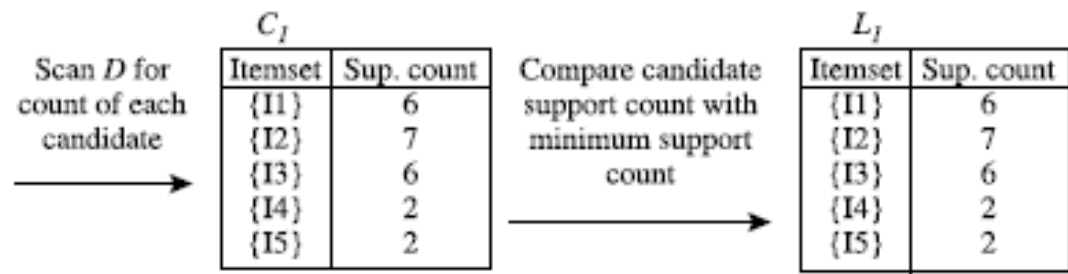


Example: Apriori Algorithm

$C_2 \rightarrow L_2$

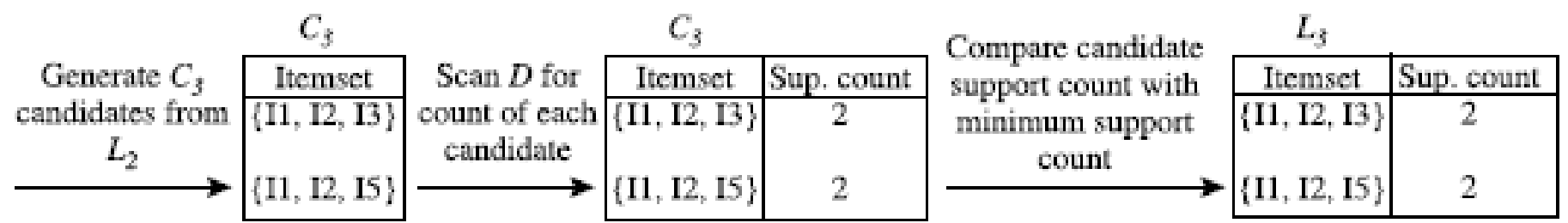


<i>TID</i>	<i>List of item_IDs</i>
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



Example: Apriori Algorithm

$C_3 \rightarrow L_3$



The Apriori algorithm for discovering frequent itemsets for mining Boolean association rules.

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D , a database of transactions;
- min_sup , the minimum support count threshold.

Output: L , frequent itemsets in D .

Method:

```
(1)  $L_1 = \text{find\_frequent\_1-itemsets}(D)$ ;  
(2) for ( $k = 2; L_{k-1} \neq \phi; k++$ ) {  
(3)    $C_k = \text{apriori\_gen}(L_{k-1})$ ;  
(4)   for each transaction  $t \in D$  { // scan  $D$  for counts  
(5)      $C_t = \text{subset}(C_k, t)$ ; // get the subsets of  $t$  that are candidates  
(6)     for each candidate  $c \in C_t$   
(7)        $c.\text{count}++$ ;  
(8)   }  
(9)    $L_k = \{c \in C_k | c.\text{count} \geq min\_sup\}$   
(10) }  
(11) return  $L = \cup_k L_k$ ;
```

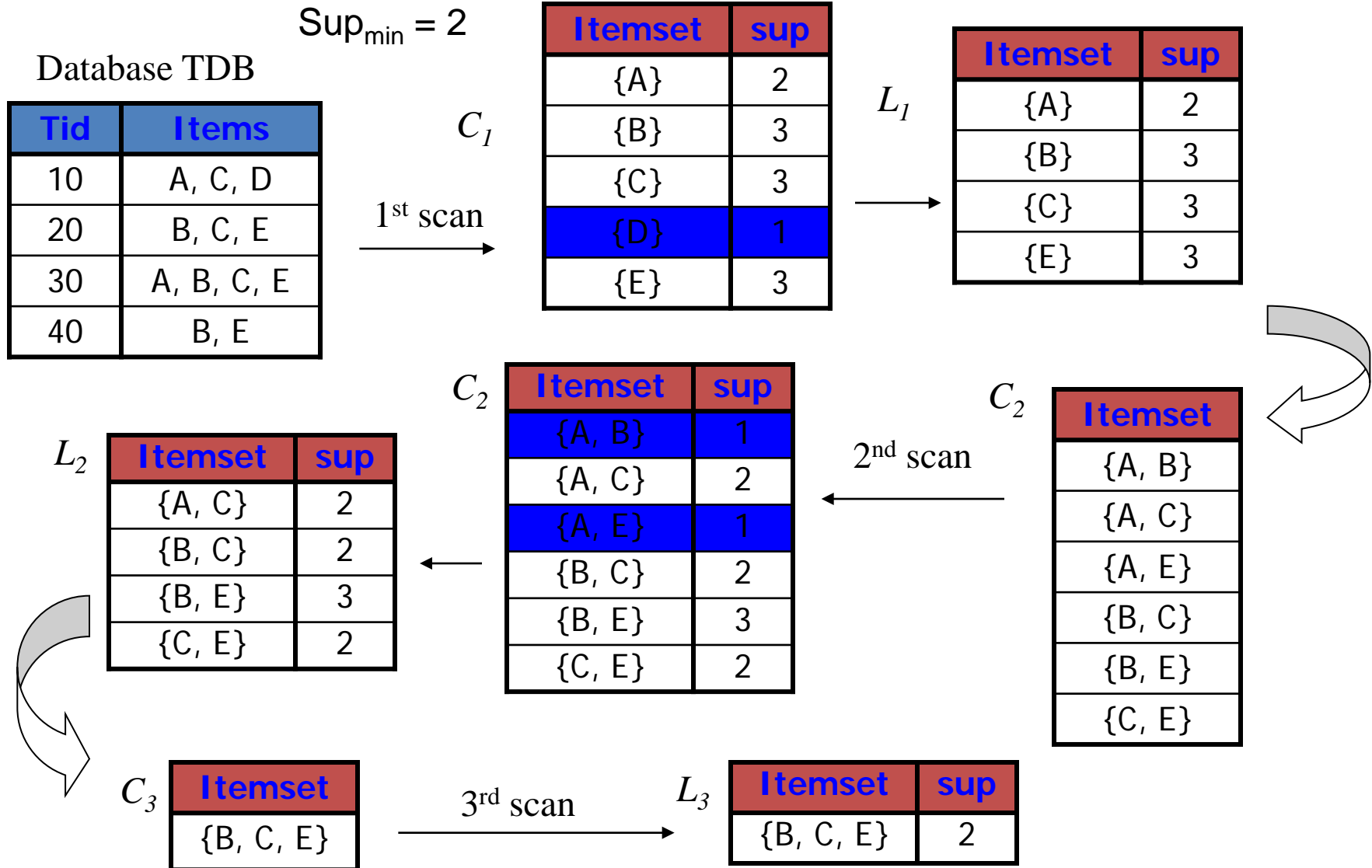
procedure $\text{apriori_gen}(L_{k-1}:\text{frequent } (k-1)\text{-itemsets})$

```
(1) for each itemset  $l_1 \in L_{k-1}$   
(2)   for each itemset  $l_2 \in L_{k-1}$   
(3)     if ( $l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \dots \wedge l_1[k-2] = l_2[k-2] \wedge l_1[k-1] < l_2[k-1]$ ) then {  
(4)        $c = l_1 \bowtie l_2$ ; // join step: generate candidates  
(5)       if  $\text{has\_infrequent\_subset}(c, L_{k-1})$  then  
(6)         delete  $c$ ; // prune step: remove unfruitful candidate  
(7)       else add  $c$  to  $C_k$ ;  
(8)     }  
(9) return  $C_k$ ;
```

procedure $\text{has_infrequent_subset}(c:\text{candidate } k\text{-itemset}$;

```
   $L_{k-1}:\text{frequent } (k-1)\text{-itemsets}$ ); // use prior knowledge  
(1) for each  $(k-1)$ -subset  $s$  of  $c$   
(2)   if  $s \notin L_{k-1}$  then  
(3)     return TRUE;  
(4) return FALSE;
```

The Apriori Algorithm—An Example



The Apriori Algorithm

- Pseudo-code:

C_k : Candidate itemset of size k

L_k : frequent itemset of size k

$L_1 = \{\text{frequent items}\};$

for ($k = 1; L_k \neq \emptyset; k++$) **do begin**

C_{k+1} = candidates generated from L_k ;

for each transaction t in database **do**

 increment the count of all candidates in C_{k+1}

 that are contained in t

L_{k+1} = candidates in C_{k+1} with min_support

end

return $\cup_k L_k$;

Important Details of Apriori

- How to generate candidates?
 - Step 1: self-joining L_k
 - Step 2: pruning
- How to count supports of candidates?
- Example of Candidate-generation
 - $L_3 = \{abc, abd, acd, ace, bcd\}$
 - Self-joining: $L_3 * L_3$
 - $abcd$ from abc and abd
 - $acde$ from acd and ace
 - Pruning:
 - $acde$ is removed because ade is not in L_3
 - $C_4 = \{abcd\}$

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}
insert into C_k
select $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$
from $L_{k-1} p, L_{k-1} q$
where $p.item_1=q.item_1, \dots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} <$
 $q.item_{k-1}$
- Step 2: pruning
forall *itemsets* c in C_k do
 forall *(k-1)-subsets* s of c do
 if (s is not in L_{k-1}) then delete c from C_k

Generating Association Rules from Frequent Itemsets

$$\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\text{support_count}(A \cup B)}{\text{support_count}(A)}$$

- For each frequent itemset l , generate all nonempty subsets of l .
- For every nonempty subset s of l , output the rule " $s \Rightarrow (l - s)$ " if $\frac{\text{support_count}(l)}{\text{support_count}(s)} \geq \text{min_conf}$, where min_conf is the minimum confidence threshold.

Example:

Generating association rules

- frequent itemset $I = \{I1, I2, I5\}$

$$I1 \wedge I2 \Rightarrow I5,$$

$$\text{confidence} = 2/4 = 50\%$$

$$I1 \wedge I5 \Rightarrow I2,$$

$$\text{confidence} = 2/2 = 100\%$$

$$I2 \wedge I5 \Rightarrow I1,$$

$$\text{confidence} = 2/2 = 100\%$$

$$I1 \Rightarrow I2 \wedge I5,$$

$$\text{confidence} = 2/6 = 33\%$$

$$I2 \Rightarrow I1 \wedge I5,$$

$$\text{confidence} = 2/7 = 29\%$$

$$I5 \Rightarrow I1 \wedge I2,$$

$$\text{confidence} = 2/2 = 100\%$$

<i>TID</i>	<i>List of item.IDs</i>
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

- If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules above are output, because these are the only ones generated that are strong.

Summary

- Association Analysis
- Mining Frequent Patterns
- Association and Correlations
- Apriori Algorithm

References

- Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, 2006, Elsevier
- Efraim Turban, Ramesh Sharda, Dursun Delen, Decision Support and Business Intelligence Systems, Ninth Edition, 2011, Pearson.