Frequent Pattern Mining

Agenda

- Frequent Pattern : market basket analysis
- Association Rules
- Apriori Algorithm
- FP-growth: FP-tree
- FP Mining with Vertical Data Format
- Lift : Correlation measure

Basic Concepts

- **Frequent Pattern:** a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set

- **Goal:** finding inherent regularities in data
  - What products were often purchased together?— Milk and Bread?!
  - 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:**
  - A typical example of frequent itemset mining is **market basket analysis.**
  - cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

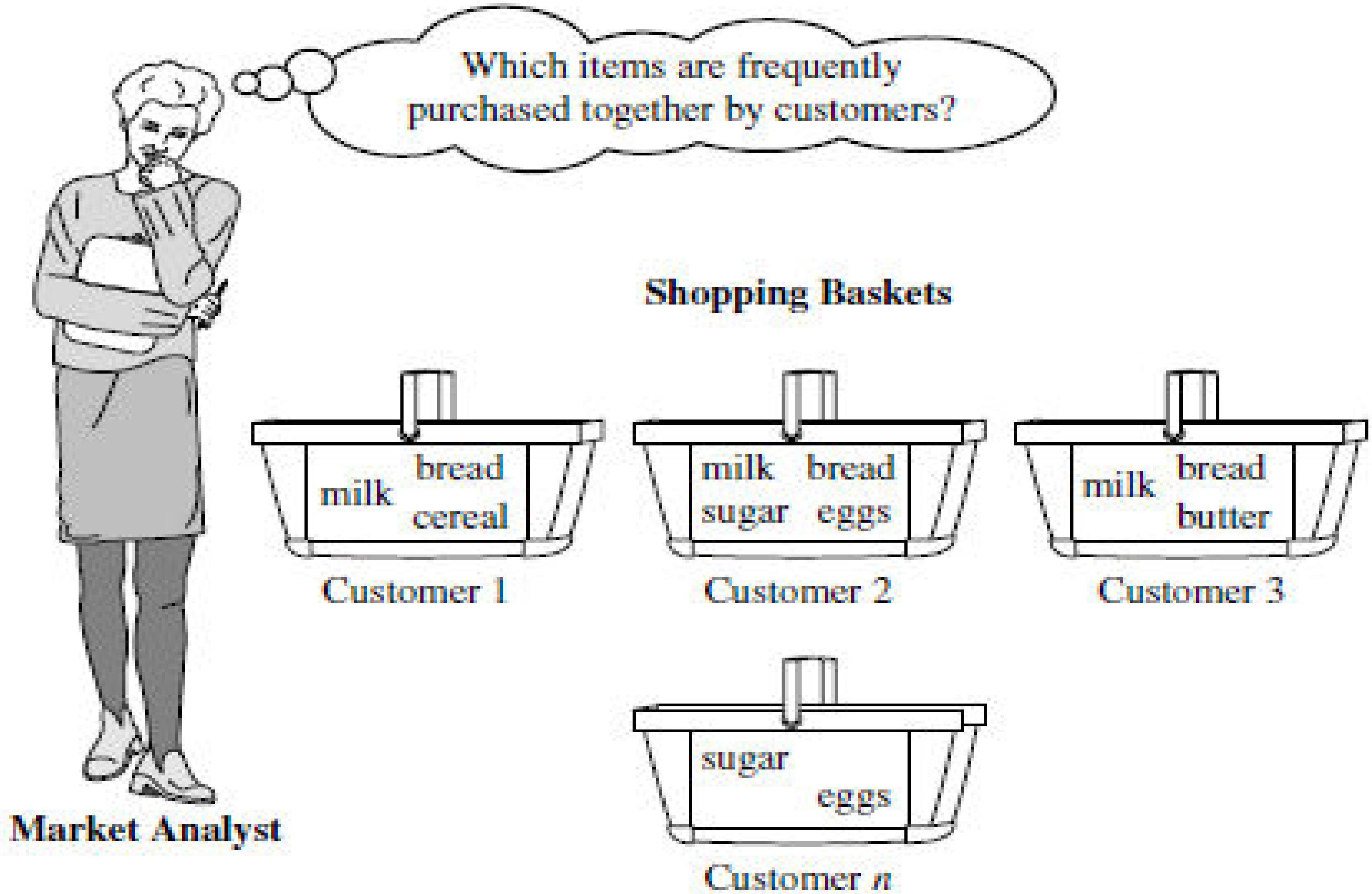
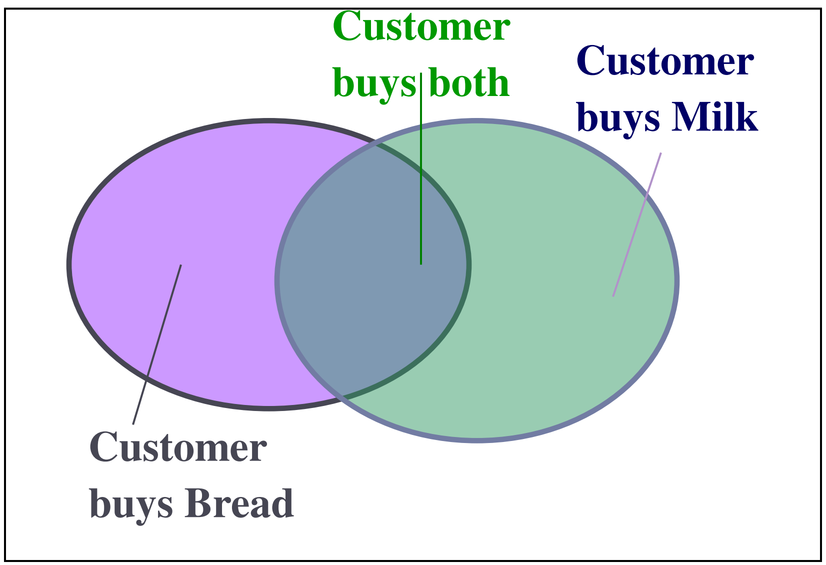


Figure 1 Market basket analysis.

First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining (Cited by 25688)

Frequent Patterns

Tid	Items bought
10	Milk, Bread, Cookies, Juice
20	Milk, Juice
30	Milk, Eggs
40	Bread, Cookies, Coffee

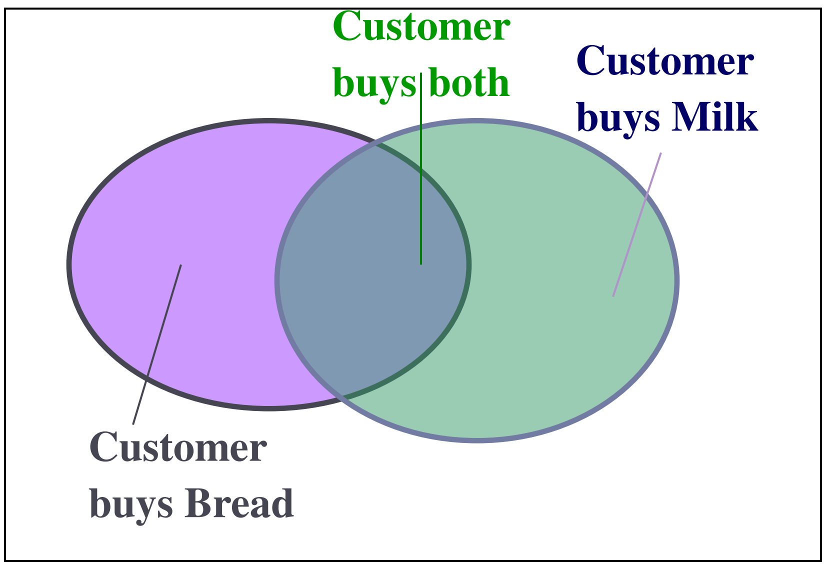


- Itemset $X = \{x_1, \ldots, x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - support, *s*, probability that a transaction contains $X \cup Y$
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

- An itemset X is **frequent** if X's support is no less than a *minsup* threshold

Association Rules

Tid	Items bought
10	Milk, Bread, Cookies, Juice
20	Milk, Juice
30	Milk, Eggs
40	Bread, Cookies, Coffee



- Find all the rules *X* → *Y* with minimum support and confidence threshold
  - **support** , s, probability that a transaction contains X ∪ Y
  - **confidence**, c, conditional probability that a transaction having X also contains Y

Let Milk → Juice and Bread → Juice

- The support of Milk→ Juice = 50% **(2/4)**
- The Conf of Milk → Juice = 66.7% **(2/3)**
- The support of Bread → Juice = 25% **?**
- The Conf of Bread → Juice = 50% **?**

- Rules that satisfy both minsup and minconf are called **strong rules**

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., **$\{a_1, \ldots, a_{100}\}$** contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 =$ **$1.27*10^{30}$ sub-patterns!**

- Solution: *Mine* **closed patterns** *and* **max-patterns** *instead*

- An itemset X is **closed** if X is *frequent* and there exists *no super-pattern* Y $\supset$ X*, with the same support* as X

- An itemset X is a **max-pattern** if X is frequent and there exists no frequent super-pattern Y $\supset$ X

- Closed pattern is a lossless compression of freq. patterns
  - → Reducing the number of patterns and rules

Closed Patterns and Max-Patterns

Example

- $DB = \{<a_1, \ldots, a_{100}>, < a_1, \ldots, a_{50}>\}$
- Min_sup=1

- What is the set of **closed itemset?**
  - → <a1, …, a100>: 1
  - → < a1, …, a50>: 2

- What is the set of **max-pattern?**
  - → <a1, …, a100>: 1

Computational Complexity

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is sensitive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: MN where M: # distinct items, and N: max length of transactions

Apriori: Concepts and Principle

- The **downward closure** property of frequent patterns
  - → Any subset of a frequent itemset must be frequent
  - → If {Milk, Bread, Juice} is frequent, so is {Milk, Bread}
  - → i.e., every transaction having {Milk, Bread, Juice} also contains {Milk, Bread}
- **Apriori pruning principle:** If there is **any** itemset which is infrequent, its superset should not be generated/tested

Apriori Principle

- Initially, scan DB once to get frequent 1-itemset
- **Generate** length (k+1) **candidate** itemsets from length k frequent itemsets
- **Test** the candidates against DB
- Terminate when no frequent or candidate set can be generated

Apriori: Example

$Sup_{min} = 2$

Database

Tid	Items
10	I1, I3, I4
20	I2, I3, I5
30	I1, I2, I3, I5
40	I2, I5

1^{st} scan

C_1

Itemset	sup
{I1}	2
{I2}	3
{I3}	3
{I4}	1
{I5}	3

L_1

Itemset	sup
{I1}	2
{I2}	3
{I3}	3
{I5}	3

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I5}
{I2, I3}
{I2, I5}
{I3, I5}

2^{nd} scan

C_2

Itemset	sup
{I1, I2}	1
{I1, I3}	2
{I1, I5}	1
{I2, I3}	2
{I2, I5}	3
{I3, I5}	2

L_2

Itemset	sup
{I1, I3}	2
{I2, I3}	2
{I2, I5}	3
{I3, I5}	2

C_3

Itemset
{I2, I3, I5}

3^{rd} scan

L_3

Itemset	sup
{I2, I3, I5}	2

Apriori Algorithm

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; 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 $\cup_k L_k$;

Candidate Generation

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning

Join L_k p with L_k q, as follows:

insert into C_{k+1}

select $\{p.item_i\}_{\{1,...,k-1\}}$, $p.item_k$, $q.item_k$

from L_k p, L_k q

where $\{p.item_i\}_{\{1,..k-1\}} = \{q.item_i\}_{\{1,..k-1\}}$ and $p.item_k < q.item_k$

Example of Candidate Generation

Suppose we have the following frequent 3-itemsets and we would like to generate the 4-itemsets candidates

- L3={{I1, I2, I3} , {I1, I2, I4}, {I1, I3, I4}, {I1, I3, I5}, {I2,I3,I4}}
- Self-joining: L3*L3 gives:
  {I1,I2,I3,I4} from {I1, I2, I3} , {I1, I2, I4}, and {I2,I3,I4}
  {I1,I3,I4,I5} from {I1, I3, I4} and {I1, I3, I5}
- Pruning: {I1,I3,I4,I5} is removed because {I1,I4,I5} is not in L3
- **C4 = {I1,I2,I3,I4}**

Generating Association Rules

- Once the frequent itemsets have been found, it is straightforward to generate **strong** association rules that satisfy:

  - → **minimum** support
  - → **minimum** confidence

- Relation between support and confidence:

$$\textbf{Confidence}(A \Rightarrow B) = P(B \mid A) = \frac{\textbf{support\_count}(A \cup B)}{\textbf{support\_count}(A)}$$

  - → **Support_count(A∪B)** is the number of transactions containing the itemsets $A \cup B$
  - → **Support_count(A)** is the number of transactions containing the itemset A.

Generating Association Rules

- For each frequent itemset **L**, generate all non empty subsets of **L**
- *For* every non empty subset **S** of **L**, output the rule:

$$\mathbf{S \Rightarrow (L\text{-}S)}$$

If (support_count(L)/support_count(S)) >= min_conf

Example

- Suppose the frequent Itemset **L={I1,I2,I5}**
- Subsets of L are: **{I1,I2}, {I1,I5},{I2,I5},{I1},{I2},{I5}**
- Association rules :

I1 $\wedge$ I2 $\Rightarrow$ I5	confidence = 2/4= 50%
I1 $\wedge$ I5 $\Rightarrow$ I2	confidence=2/2=100%
I2 $\wedge$ I5 $\Rightarrow$ I1	confidence=2/2=100%
I1 $\Rightarrow$ I2 $\wedge$ I5	confidence=2/6=33%
I2 $\Rightarrow$ I1 $\wedge$ I5	confidence=2/7=29%
I5 $\Rightarrow$ I2 $\wedge$ I2	confidence=2/2=100%

If the minimum confidence =70%

Transactional Database

TID	List of item IDS
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

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Improving the Efficiency of Apriori

- **Major computational challenges**
  - Huge number of candidates
  - Multiple scans of transaction database
  - Tedious workload of support counting for candidates

- **Improving Apriori: general ideas**
  - Hash-based technique
  - Transaction reduction
  - Partitioning
  - Sampling

Exercises

- A database has five transactions. Let *min sup D 60% and min conf D 80%.*

TID	*items_bought*
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y }
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I, E}

- Find all frequent itemsets using Apriori.
- List all the *strong association rules.*

FP-growth: Frequent Pattern-Growth

- Adopts a divide and conquer strategy
- Compress the database representing frequent items into a **frequent –pattern tree** or **FP-tree**
  - → Retains the itemset association information
- Divid the compressed database into a set of conditional databases, each associated with one frequent item
- Mine each such databases separately

Example: FP-growth

- The first scan of data is the same as Apriori
- Derive the set of frequent 1-itemsets
- Let min-sup=2
- Generate a set of ordered items

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2

Transactional Database

TID	List of item IDS
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

Construct the FP-Tree

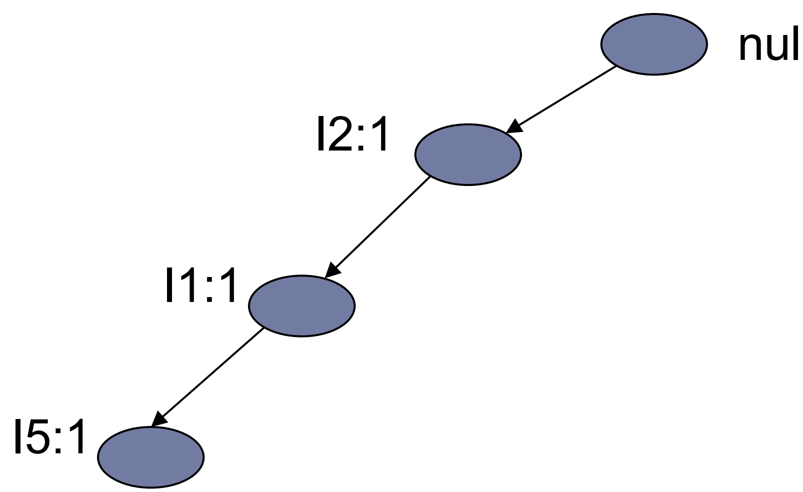
Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

- Create a branch for each transaction
- Items in each transaction are processed in order

1- Order the items T100: {I2,I1,I5}
2- Construct the first branch: <I2:1>, <I1:1>,<I5:1>

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



Construct the FP-Tree

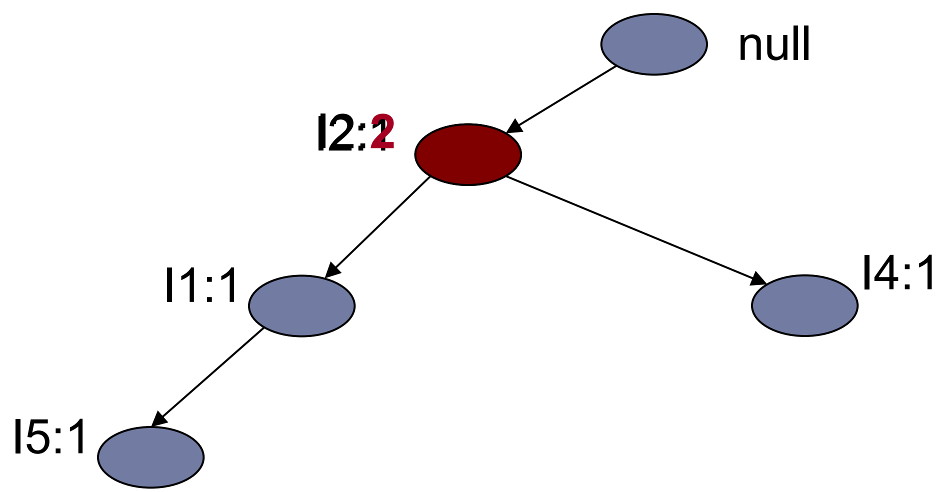
Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

- Create a branch for each transaction
- Items in each transaction are processed in order

1- Order the items T200: {I2,I4}
2- Construct the second branch: <I2:1>, <I4:1>

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



Construct the FP-Tree

Transactional Database

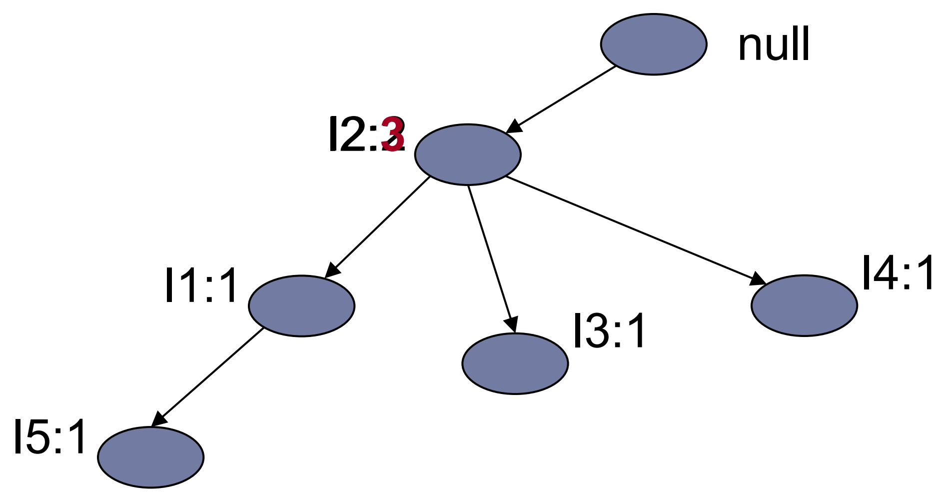
TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

- Create a branch for each transaction
- Items in each transaction are processed in order

1- Order the items T300: {I2,I3}

2- Construct the third branch: <I2:2>, <I3:1>

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



Construct the FP-Tree

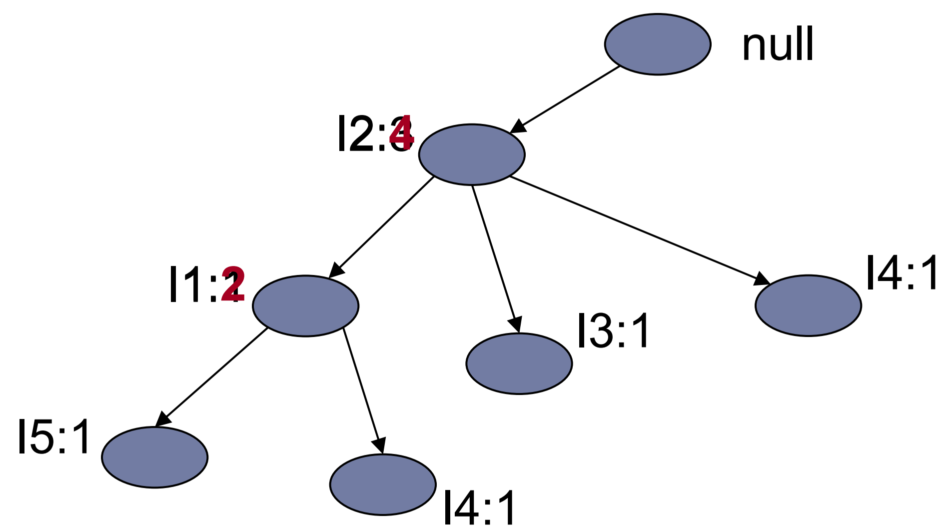
Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

- Create a branch for each transaction
- Items in each transaction are processed in order

1- Order the items T400: {I2,I1,I4}
2- Construct the fourth branch: <I2:3>, <I1:1>,<I4:1>

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



Construct the FP-Tree

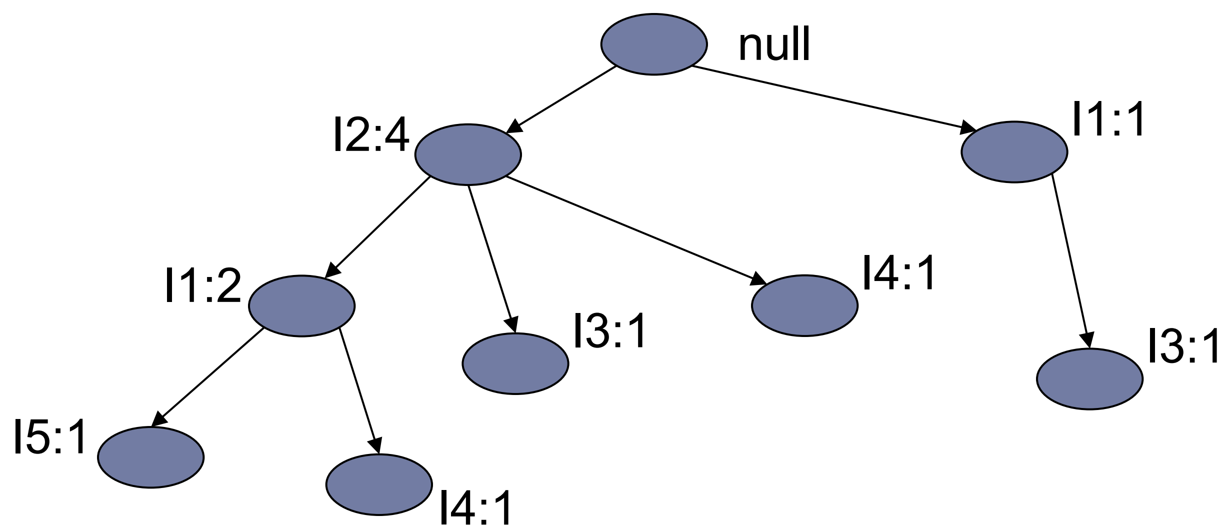
Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

- Create a branch for each transaction
- Items in each transaction are processed in order

1- Order the items T400: {I1,I3}
2- Construct the fifth branch: <I1:1>, <I3:1>

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2

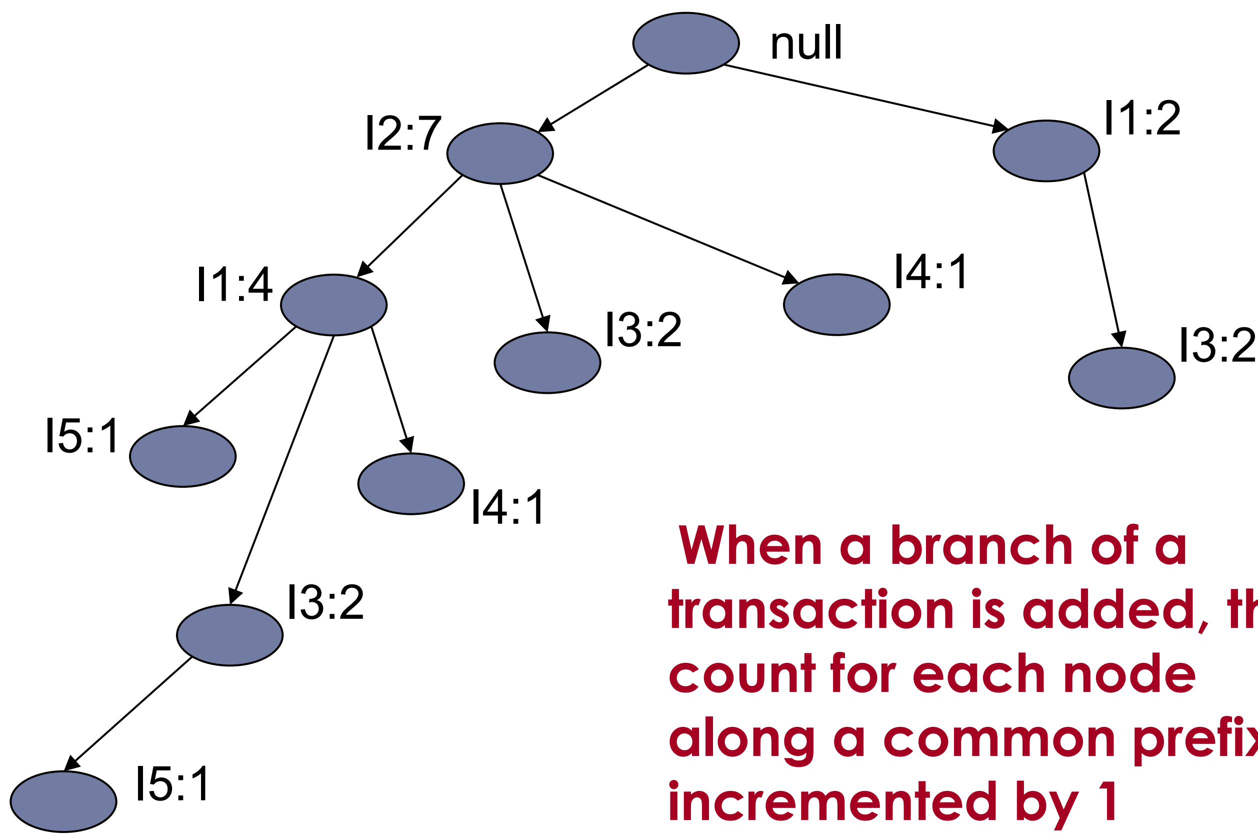


Construct the FP-Tree

Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

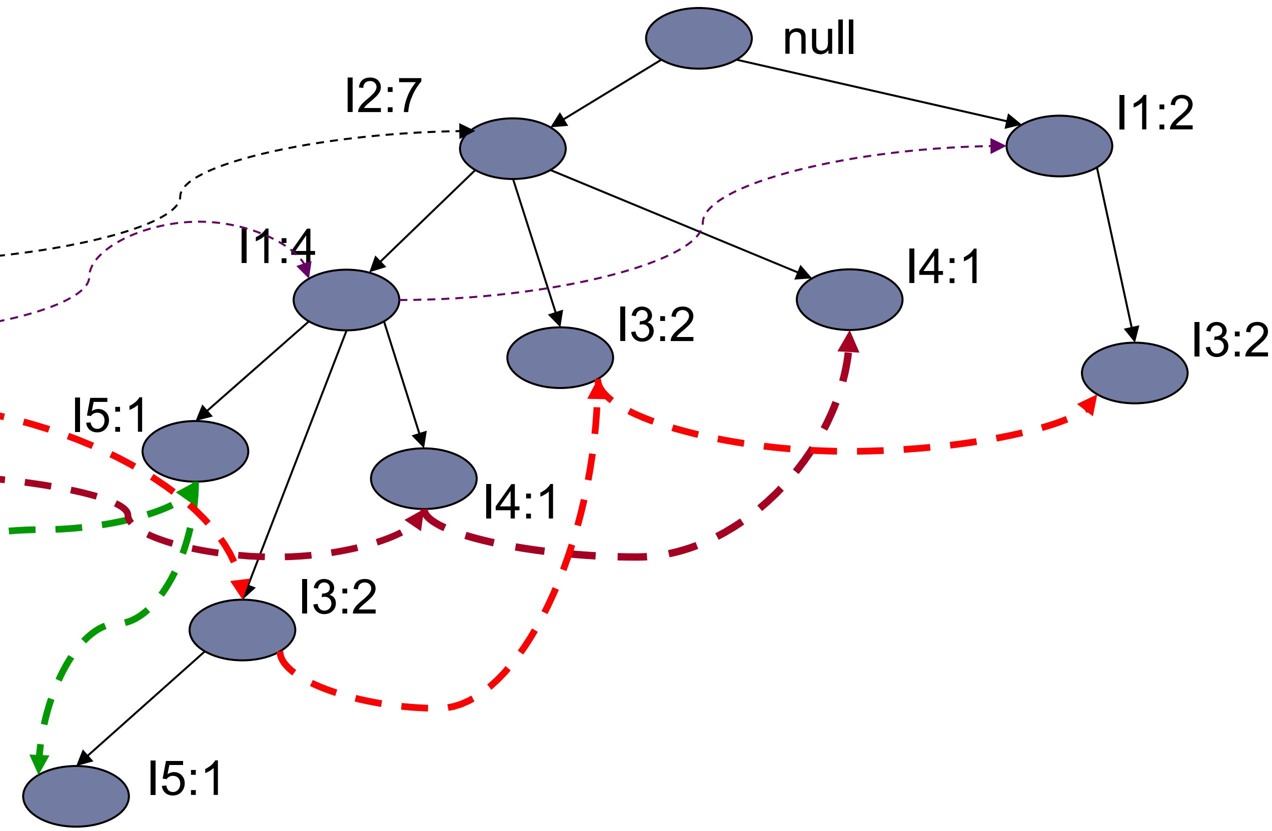
Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



When a branch of a transaction is added, the count for each node along a common prefix is incremented by 1

Construct the FP-Tree

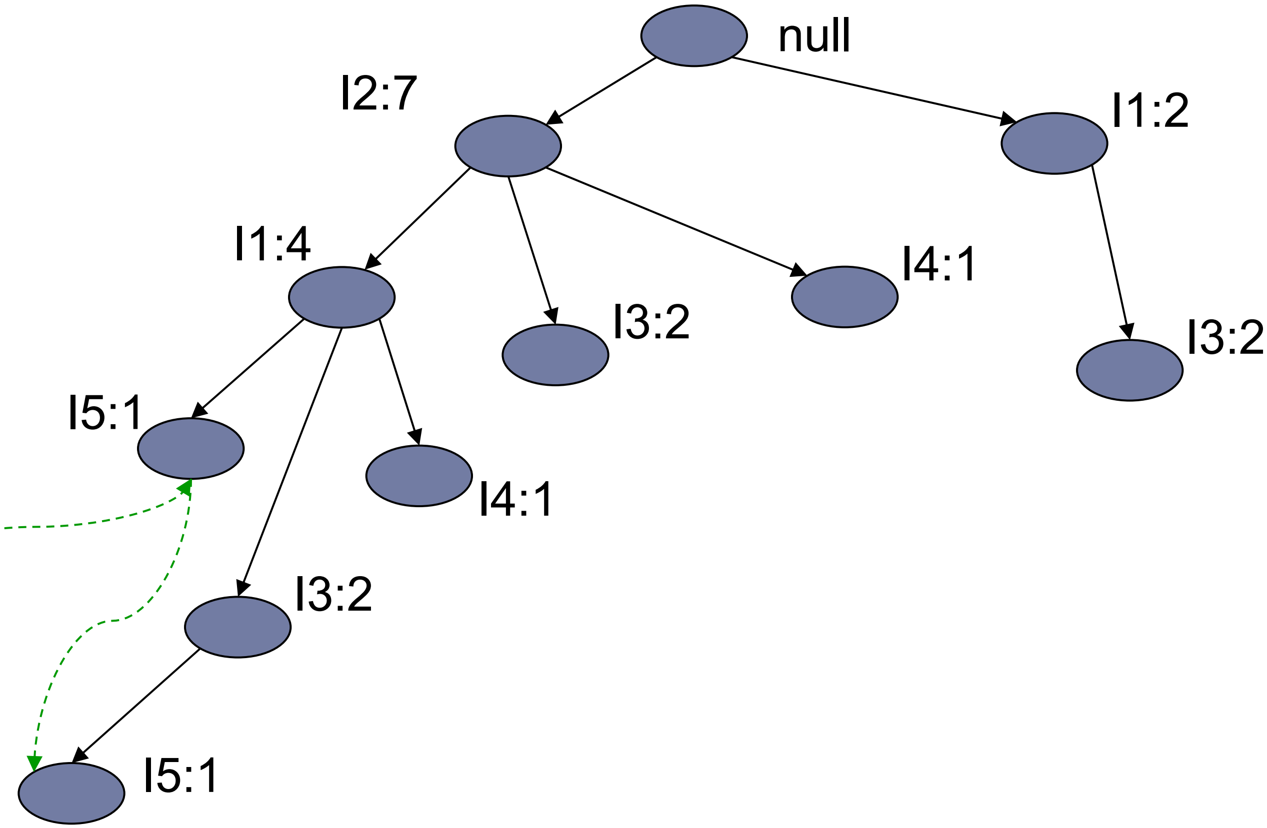
Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



The problem of mining frequent patterns in databases is transformed to that of mining the FP-tree

Construct the FP-Tree

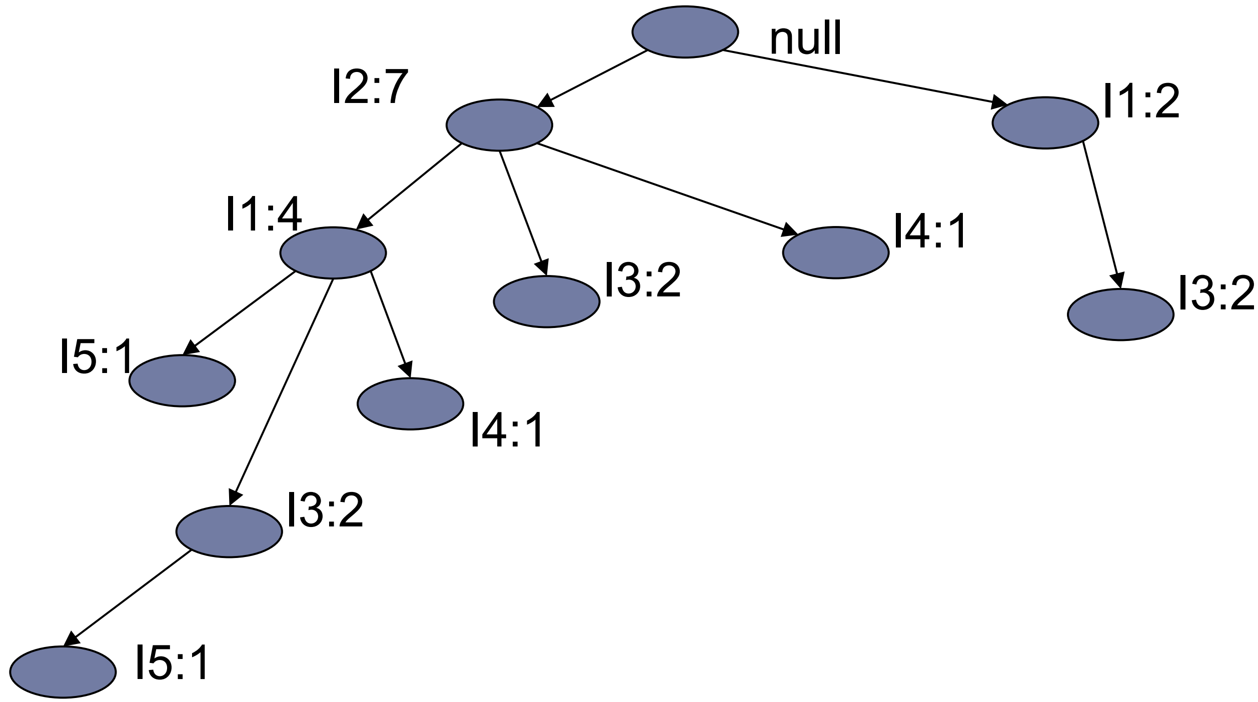
Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



- **Occurrences of I5:** <I2,I1,I5> and <I2,I1,I3,I5>
- **Two prefix Paths** <I2, I1: 1> and <I2,I1,I3: 1>
- **Conditional FP tree contains only** <I2: 2, I1: 2>, I3 is not considered because its support count of 1 is less than the minimum support count.
- **Frequent patterns** {I2,I5:2}, {I1,I5:2},{I2,I1,I5:2}

Construct the FP-Tree

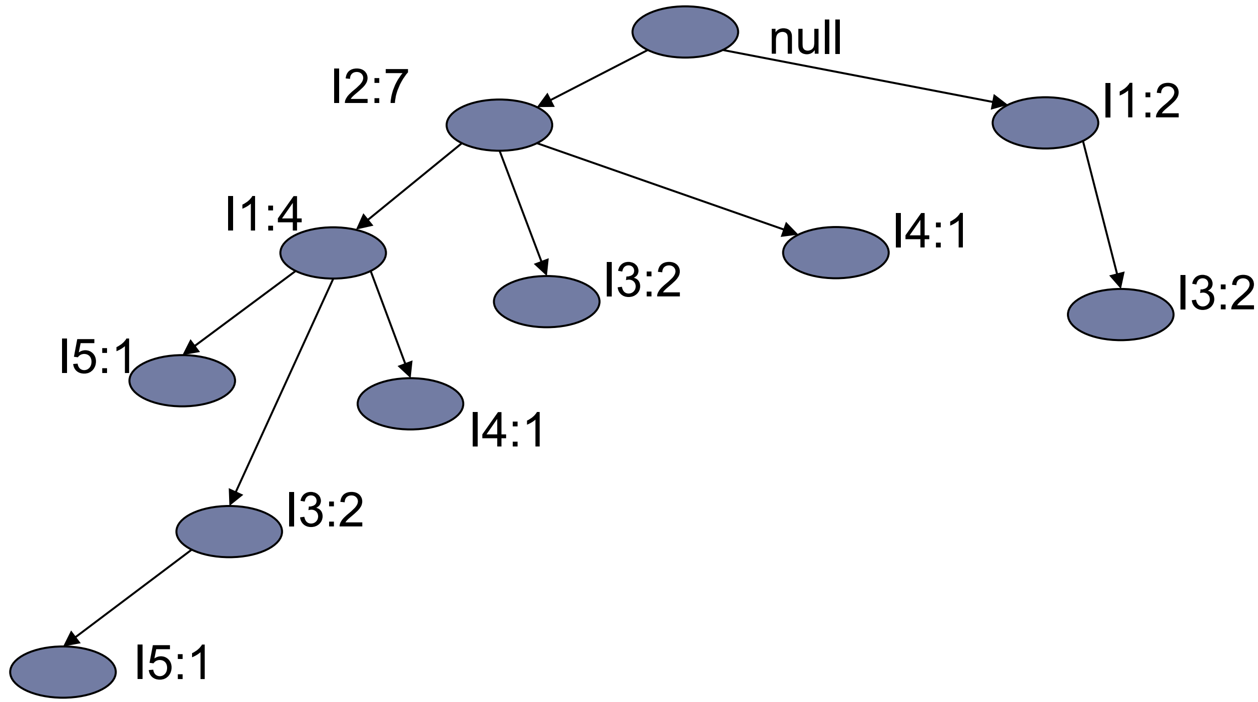
Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



TID	Conditional Pattern Base	Conditional FP-tree
I5	{{I2,I1:1},{I2,I1,I3:1}}	<I2:2,I1:2>
I4	{{I2,I1:1},{I2,1}}	<I2:2>
I3	{{I2,I1:2},{I2:2}, {I1:2}}	<I2:4,I1:2>,**<I1:2>**
I1	{I2,4}	<I2:4>

Construct the FP-Tree

Item ID	**Support count**
I2	7
I1	6
I3	6
I4	2
I5	2



TID	**Conditional FP-tree**	**Frequent Patterns Generated**
I5	<I2:2,I1:2>	{I2,I5:2}, {I1,I5:2},{I2,I1,I5:2}
I4	<I2:2>	{I2,I4:2}
I3	<I2:4,I1:2>,**<I1:2>**	{I2,I3:4},{I1,I3:4},**{I2,I1,I3:2}**
I1	<I2:4>	{I2,I1:4}

FP-growth properties

- FP-growth transforms the problem of finding long frequent patterns to searching for shorter once recursively and concatenating the suffix
- It uses the least frequent suffix offering a good selectivity
- It reduces the search cost
- **If the tree does not fit into main memory, partition the database**
- Efficient and scalable for mining both long and short frequent patterns

FP Mining with Vertical Data Format

- Both **Apriori** and **FP-growth** use **horizontal data format**

TID	List of item IDS
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

- Alternatively data can also be represented in **vertical format**

itemset	TID_set
I1	{T100,T400,T500,T700,T800,T900}
I2	{T100,T200,T300,T400,T600,T800,T900}
I3	{T300,T500,T600,T700,T800,T900}
I4	{T200,T400}
I5	{T100,T800}

Example

- Transform the horizontally formatted data to the vertical format by scanning the database once

TID	List of item IDS
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

→

itemset	TID_set
I1	{T100,T400,T500,T700,T800,T900}
I2	{T100,T200,T300,T400,T600,T800,T900}
I3	{T300,T500,T600,T700,T800,T900}
I4	{T200,T400}
I5	{T100,T800}

- The support count of an itemset is simply **the length of the TID**_set of the itemset

Example ...

Frequent 1-itemsets in vertical format

min_sup=2

itemset	**TID_set**
I1	{T100,T400,T500,T700,T800,T900}
I2	{T100,T200,T300,T400,T600,T800,T900}
I3	{T300,T500,T600,T700,T800,T900}
I4	{T200,T400}
I5	{T100,T800}

- The frequent k-itemsets can be used to construct the candidate (k+1)-itemsets based on the Apriori property

Frequent 2-itemsets in vertical format

itemset	**TID_set**
{I1,I2}	{T100,T400,T800,T900}
{I1,I3}	{T500,T700,T800,T900}
{I1,I4}	{T400}
{I1,I5}	{T100,T800}
{I2,I3}	{T300,T600,T800,T900}
{I2,I4}	{T200,T400}
{I2,I5}	{T100,T800}
{I3,I5}	{T800}

Example ...

Frequent 3-itemsets in vertical format

min_sup=2

itemset	TID_set
{I1,I2,I3}	{T800,T900}
{I1,I2,I5}	{T100,T800}

- This process repeats, with k incremented by 1 each time, until no frequent items or no candidate itemsets can be found
- **Properties of mining with vertical data format**
  - Take the advantage of the Apriori property in the generation of candidate (k+1)-itemset from k-itemsets
  - No need to scan the database to find the support of (k+1) itemsets, for k>=1
  - The TID_set of each k-itemset carries the complete information required for counting such support
  - The TID-sets can be quite long, hence expensive to manipulate

Strong Rules Are Not Necessarily Interesting

- Whether a rule is interesting or not can be assessed either subjectively or objectively
- Objective interestingness measures can be used as one step toward the goal of finding interesting rules for the user

- **Example of a misleading "strong" association rule**
  - Analyze transactions of AllElectronics data about computer games and videos
  - Of the **10,000** transactions analyzed
    - **6,000** of the transactions include **computer games**
    - **7,500** of the transactions include **videos**
    - **4,000** of the transactions include **both**
  - Suppose that min_sup=30% and min_confidence=60%
  - The following association rule is discovered:

Buys(X, "computer games") ⇒ buys(X, "videos")[support =40%, confidence=66%]

Strong Rules Are Not Necessarily Interesting

Buys(X, "computer games") ⇒ buys(X, "videos")[support 40%, confidence=66%]

- This rule is strong but it is misleading
- The probability of purchasing videos is **75%** which is even larger than **66%**
- In fact computer games and videos are negatively associated because the purchase of one of these items actually decreases the likelihood of purchasing the other
- The confidence of a rule **A ⇒ B** can be deceiving
  - It is only an estimate of the conditional probability of itemset **B** given itemset **A**.
  - It does not measure the real strength of the correlation implication between **A** and **B**
- Need to use **Correlation Analysis**

From Association to Correlation Analysis

A misleading "strong" association rule. Suppose we are interested in analyzing transactions at *AllElectronics* with respect to the purchase of computer games and videos. Let *game* refer to the transactions containing computer games, and *video* refer to those containing videos. Of the 10,000 transactions analyzed, the data show that 6000 of the customer transactions included computer games, while 7500 included videos, and 4000 included both computer games and videos. Suppose that a data mining program for discovering association rules is run on the data, using a minimum support of, say, 30% and a minimum confidence of 60%. The following association rule is discovered:

$$buys(X, \text{"computer games"}) \Rightarrow buys(X, \text{"videos"})$$
$$[support = 40\%, confidence = 66\%]. \tag{6.6}$$

Rule (6.6) is a strong association rule and would therefore be reported, since its support value of $\frac{4000}{10,000} = 40\%$ and confidence value of $\frac{4000}{6000} = 66\%$ satisfy the minimum support and minimum confidence thresholds, respectively. However, Rule (6.6) is misleading because the probability of purchasing videos is 75%, which is even larger than 66%. In fact, computer games and videos are negatively associated because the purchase of one of these items actually decreases the likelihood of purchasing the other. Without fully understanding this phenomenon, we could easily make unwise business decisions based on Rule (6.6). ■

From Association to Correlation Analysis

- Use **Lift**, a simple correlation measure

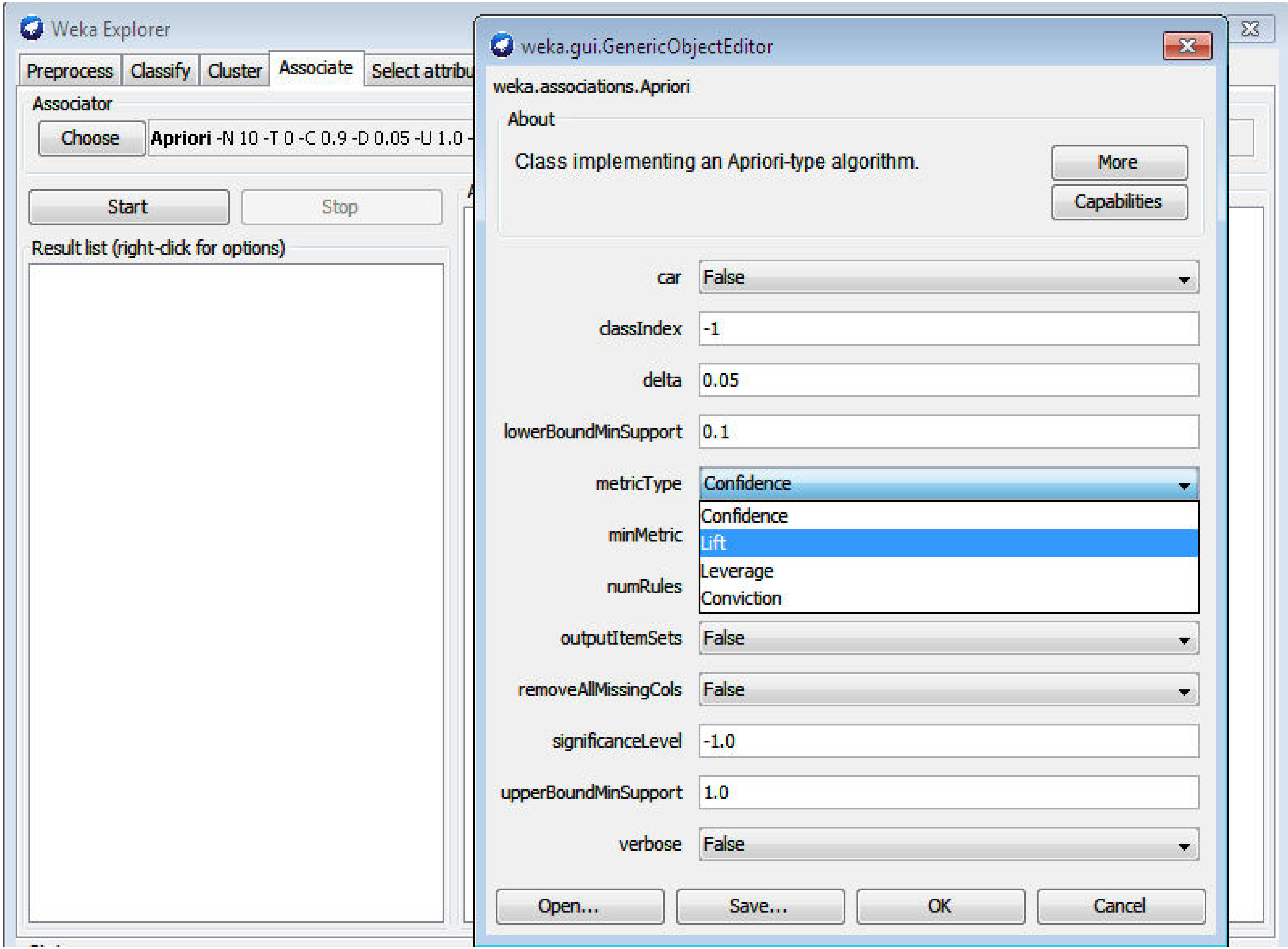
- The occurrence of itemset **A** is independent of the occurrence of itemset **B** if **P(A∪B)=P(A)P(B)**, otherwise itemsets **A** and **B** are dependent and correlated as events

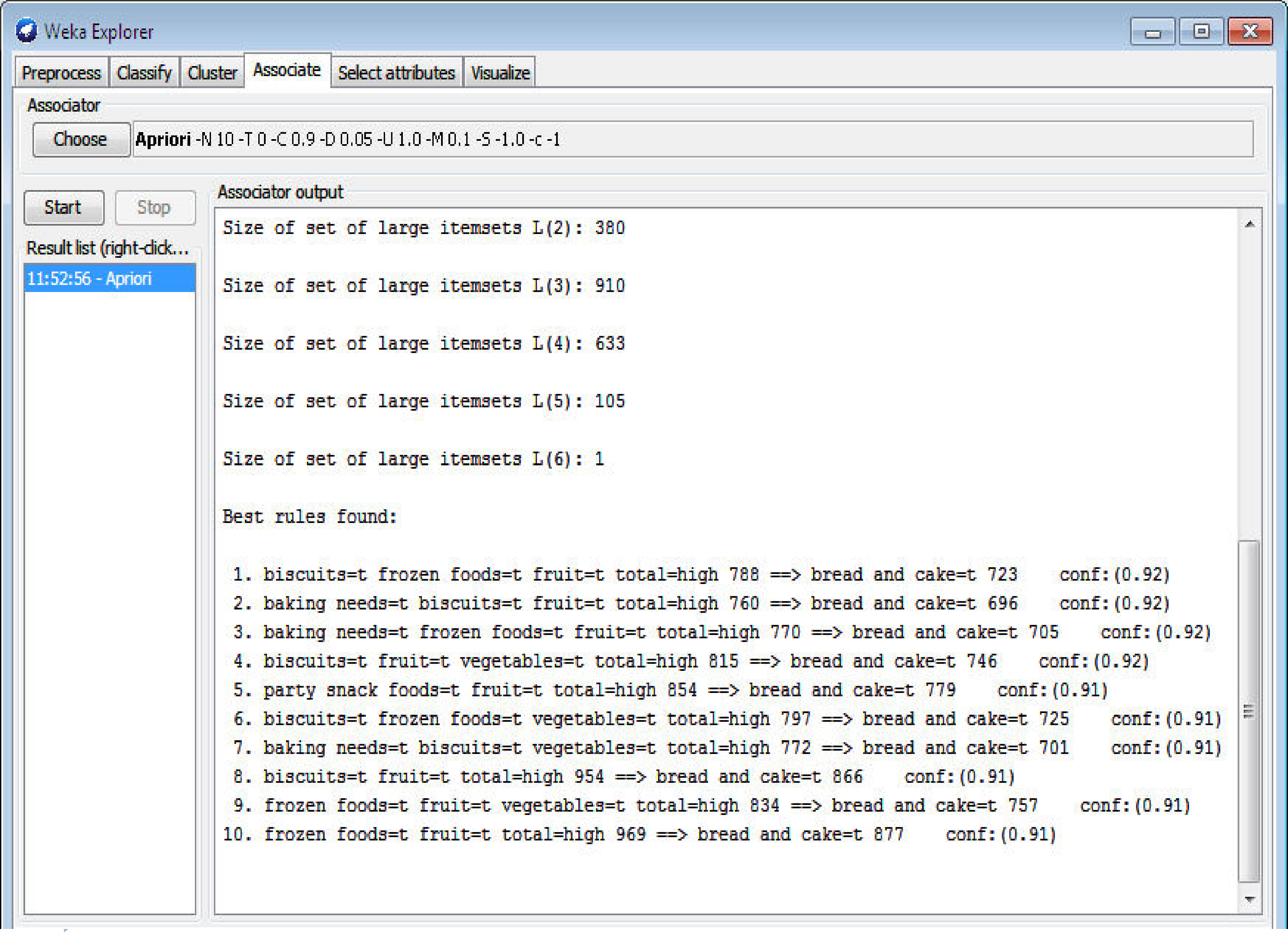
- The lift between the occurences of **A** and **B** is given by

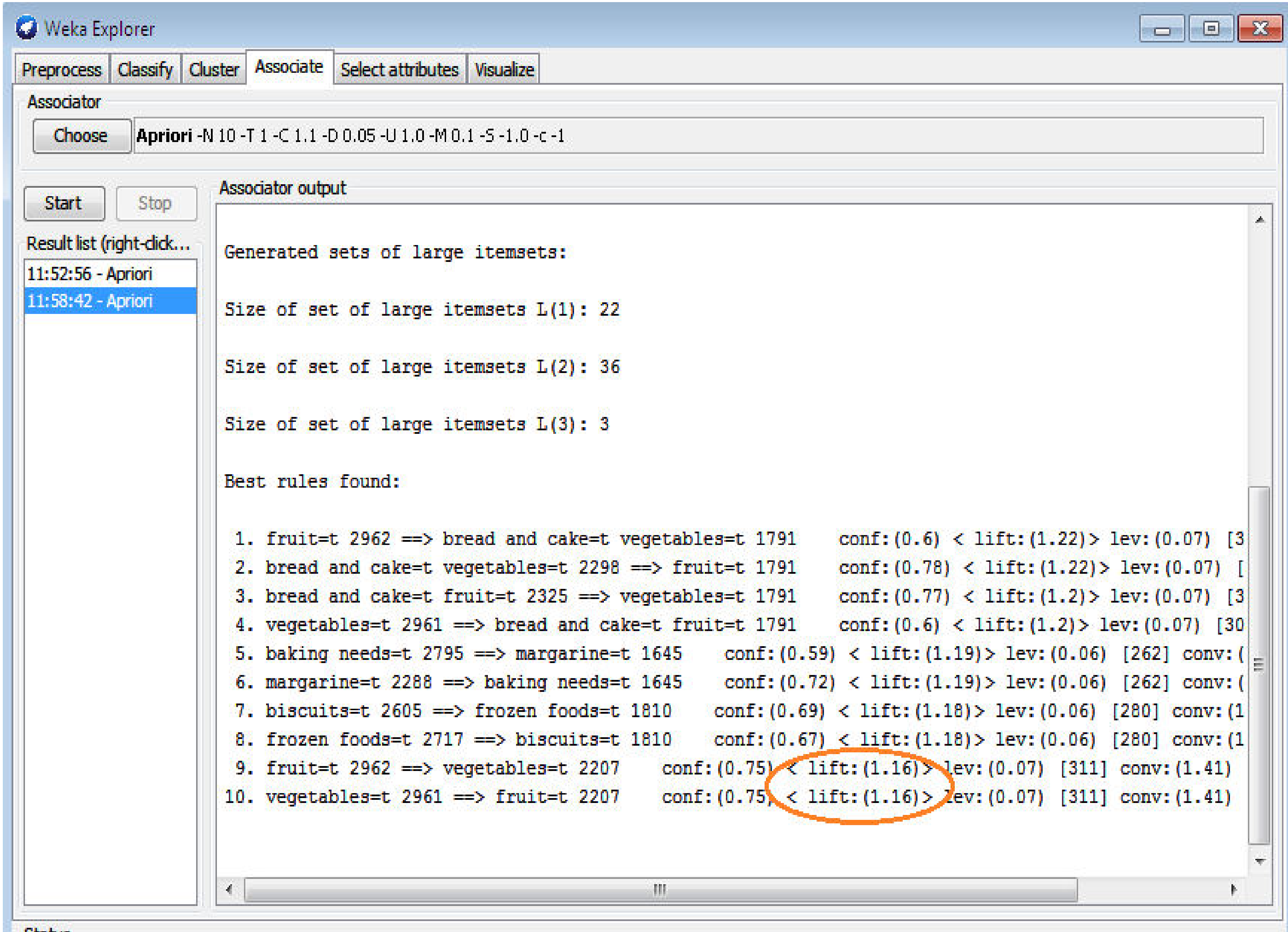
Lift(A,B)=P(A∪B)/P(A)P(B)

→ If > 1, then A and B are positively correlated (the occurrence of one implies the occurrence of the other)
→ If <1, then A and B are negatively correlated
→ If =1, then A and B are independent

- **Example: P({game, video})=0.4/(0.60 × 0.75)=0.89**


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attribu
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -
Start
Stop
Result list (right-click for options)
weka.gui.GenericObjectEditor
weka.associations.Apriori
About
Class implementing an Apriori-type algorithm.
More
Capabilities
car
False
classIndex
-1
delta
0.05
lowerBoundMinSupport
0.1
metricType
Confidence
Confidence
Lift
Leverage
Conviction
minMetric
numRules
outputItemSets
False
removeAllMissingCols
False
significanceLevel
-1.0
upperBoundMinSupport
1.0
verbose
False
Open...
Save...
OK
Cancel


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-click...
11:52:56 - Apriori
Associator output
Size of set of large itemsets L(2): 380
Size of set of large itemsets L(3): 910
Size of set of large itemsets L(4): 633
Size of set of large itemsets L(5): 105
Size of set of large itemsets L(6): 1
Best rules found:
1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723 conf:(0.92)
2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696 conf:(0.92)
3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705 conf:(0.92)
4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746 conf:(0.92)
5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779 conf:(0.91)
6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725 conf:(0.91)
7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701 conf:(0.91)
8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866 conf:(0.91)
9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757 conf:(0.91)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877 conf:(0.91)


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 1 -C 1.1 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-click...
11:52:56 - Apriori
11:58:42 - Apriori
Associator output
Generated sets of large itemsets:
Size of set of large itemsets L(1): 22
Size of set of large itemsets L(2): 36
Size of set of large itemsets L(3): 3
Best rules found:
1. fruit=t 2962 ==> bread and cake=t vegetables=t 1791 conf:(0.6) < lift:(1.22)> lev:(0.07) [3
2. bread and cake=t vegetables=t 2298 ==> fruit=t 1791 conf:(0.78) < lift:(1.22)> lev:(0.07) [
3. bread and cake=t fruit=t 2325 ==> vegetables=t 1791 conf:(0.77) < lift:(1.2)> lev:(0.07) [3
4. vegetables=t 2961 ==> bread and cake=t fruit=t 1791 conf:(0.6) < lift:(1.2)> lev:(0.07) [30
5. baking needs=t 2795 ==> margarine=t 1645 conf:(0.59) < lift:(1.19)> lev:(0.06) [262] conv:(
6. margarine=t 2288 ==> baking needs=t 1645 conf:(0.72) < lift:(1.19)> lev:(0.06) [262] conv:(
7. biscuits=t 2605 ==> frozen foods=t 1810 conf:(0.69) < lift:(1.18)> lev:(0.06) [280] conv:(1
8. frozen foods=t 2717 ==> biscuits=t 1810 conf:(0.67) < lift:(1.18)> lev:(0.06) [280] conv:(1
9. fruit=t 2962 ==> vegetables=t 2207 conf:(0.75) < lift:(1.16)> lev:(0.07) [311] conv:(1.41)
10. vegetables=t 2961 ==> fruit=t 2207 conf:(0.75) < lift:(1.16)> lev:(0.07) [311] conv:(1.41)
Status

Summary

- **Basic Concepts:** association rules, support-confident framework, closed and max patterns

- **Scalable frequent pattern mining methods**

  → Apriori (Candidate generation & test)
  → **Pattern-Growth Approach**(FPgrowth)
  → Vertical format approach

- **Interesting Patterns**
  → Correlation analysis