18CSE355T- Data Mining and Analytics

Unit-2

Association Rules

UNIT – 2

Jayapradha J
AP/CSE

Association Rules

- Association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large databases.

- Analyse and predict costumers behaviour.
- If/then statements

Example

bread => butter

buys{onions, potatoes} => buys{tomatoes}

These information's are the basics for marketing activities such as product promotion /product pricing.

Association Rules Continues.....

Understanding the buying patterns can help to increase sales in several ways.
Example:

- If there is a pair of items, X and Y, that are frequently bought together
- Both X and Y can be placed on the same shelf, so that buyers of one item would be prompted to buy the other.
- Promotional discounts could be applied to just one out of the two items.
- Advertisements on X could be targeted at buyers who purchase Y.
- X and Y could be combined into a new pro flavour's of X.

Parts of Association Rule

bread => butter[20%,45%]

- Bread: Antecedent
- Butter: Consequent
- 20%: Support
- 45%: Confidence

- Support: denotes the probability that contains both bread and butter.
- Confidence: denotes the probability that a transaction containing bread also contains butter.

Examples to calculate the Support and confidence

- Consider, in the supermarket

Total Transactions: 100

Bread: 20

So 20/100*100 = 20% [Support].

In 20 transactions, butter occurs in 9 transactions

So 9/20*100 = 45%[Confidence].

Examples to calculate the Support and confidence

- **Support:** This says how popular an itemset is, as measured by the proportion of transactions in which an itemset appears. In Table, the support of {apple} is 4 out of 8, or 50%. Itemsets can also contain multiple items. For instance, the support of {apple, beer, rice} is 2 out of 8, or 25%.

$$\text{Support}\ \{🍎\} = \frac{4}{8}$$

Transaction 1	🍎	🍺	🍚	🍗
Transaction 2	🍎	🍺	🍚	
Transaction 3	🍎	🍺		
Transaction 4	🍎	🍐		
Transaction 5	🍼	🍺	🍚	🍗
Transaction 6	🍼	🍺	🍚	
Transaction 7	🍼	🍺		
Transaction 8	🍼	🍐		

$$\text{Confidence}\ \{🍎 \rightarrow 🍺\} = \frac{\text{Support}\ \{🍎, 🍺\}}{\text{Support}\ \{🍎\}}$$

Confidence: This says how likely item Y is purchased when item X is purchased, expressed as {X -> Y}. This is measured by the proportion of transactions with item X, in which item Y also appears. In Table, the confidence of {apple -> beer} is 3 out of 4, or 75%.

Classification of Association Rules

- **Single Dimensional Association Rule**

Eg: Bread => butter

Dimension: buying(one dimension)

- **Multidimensional Association Rule**

With 2 or more predicates or dimensions

Eg: Occupation(IT),age(>22) => buys(laptop)

Dimensions must be unique it should not repeat.

- **Hybrid Dimensional Association Rule**

With repetative predicates or dimensions

Eg: Time(5’0 clock), buys(tea) => buys(biscuits)

Association Mining – Fields &Algorithms

- Web Usage Mining
- Banking
- Bio informatics
- Market Based Analysis
- Credit/Debit Card Analysis
- Product Clustering
- Catalog Design

Algorithms

- Apriori Algorithm
- Elcat Algorithm
- FP Growth Algorithm

- **Frequent patterns** are patterns (such as itemsets, subsequences, or substructures) that appear in a data set frequently.
- Eg: milk and bread.

- A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (*frequent*) ***sequential pattern***.

- If a substructure occurs frequently, it is called a (frequent) **structured pattern**.
- Eg: subgraphs, subtrees.

Frequent Itemset

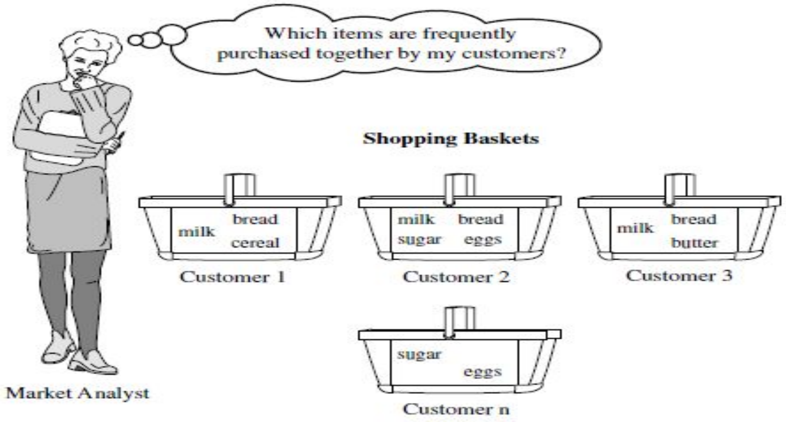
- **Itemset**
  - A collection of one or more items, e.g., {milk, bread, jam}
  - k-itemset, an itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - $\sigma(\{\text{Milk, Bread}\}) = 3$
    $\sigma(\{\text{Soda, Chips}\}) = 4$
- **Support**
  - Fraction of transactions that contain an itemset
  - $s(\{\text{Milk, Bread}\}) = 3/8$
    $s(\{\text{Soda, Chips}\}) = 4/8$
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a minsup threshold

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

Market basket Analysis

Market Basket Analysis

- Frequent item set mining leads to the discovery of associations and correlations among items in large transactional or relational data sets.
- This process analyses customer buying habits by finding associations between the different items that customers place in their “shopping baskets”.



I Market basket analysis.

Market Basket Analysis

Frequent Item sets, Closed Item sets, and Association Rules

$$support(A \Rightarrow B) = P(A \cup B)$$
$$confidence(A \Rightarrow B) = P(B|A).$$

- Rules that satisfy both a minimum support threshold (min sup) and a minimum confidence threshold (min conf) are called strong.
- A set of items is referred to as an **item set**.
- The occurrence frequency of an item set is the number of transactions that contain the itemset.
- If the relative support of an item set I satisfies a pre-specified minimum support threshold (i.e., the absolute support of I satisfies the corresponding minimum support count threshold), then I is a frequent item set.

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

Frequent Item sets, Closed Item sets, and Association Rules

In general, association rule mining can be viewed as a two-step process:

1. Find all frequent item sets: 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

Frequent Item sets, Closed Item sets, and Association Rules

- **Maximal Itemset:** An itemset is maximal frequent if none of its supersets are frequent.
- **Closed Itemset:**An itemset is closed if none of its immediate supersets have same support count same as Itemset.
- **K- Itemset:**Itemset which contains K items is a K-itemset. So it can be said that an itemset is frequent if the corresponding support count is greater than minimum support count.

Maximal & Closed Frequent Item set

Illustration

TID	Items in the Transactions
T1	{A,B,C,D}
T2	{A,D}
T3	{A,E}
T4	{C,E}

{A} is closed because none of its supersets have the same support as itself
But {A} is not maximal because {A,D} is a superset of {A} and is frequent.

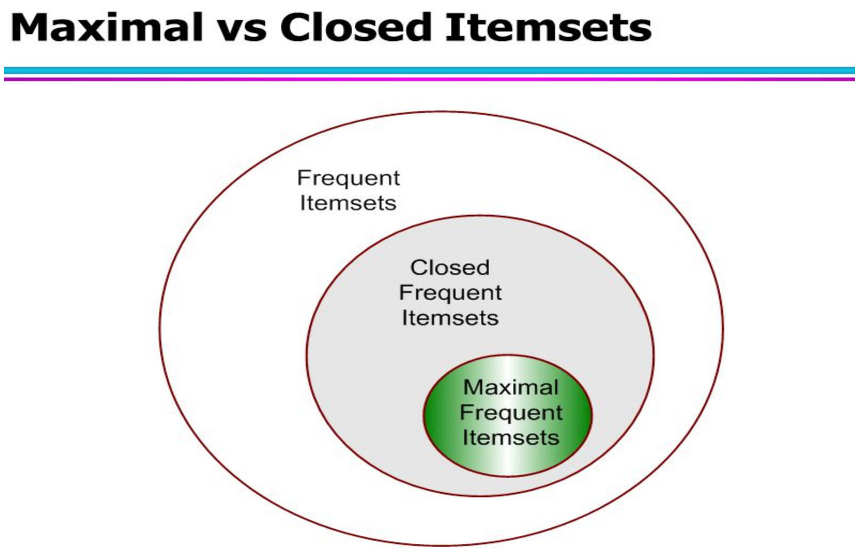
Now find the remaining frequent itemsets.

{A} - 3	{C,D} - 1
{B} - 1	{C,E} - 1
{C} -2	{D,E} - 0
{D} -2	{A,B,C} - 1
{E} - 2	{A,B,D} - 1
{A,B}- 1	{A,B,E} - 0
{A,C} - 1	{B,C,D} – 1
{A,D} -2	{B,C,E} - 0
{A,E} - 1	{C,D,E} - 0
{B,C} -1	{A,B,C,D}- 1
{B,D} - 0	{A,B,C,E}- 0
{B,E} - 0	{B,C,D,E}- 0
	{A,B,C,D,E}- 0

{{A}, {C} , {E}, {A,D} are closed frequent itemsets.
Also {C},{E} and {A,D} are maximal frequent itemset

Why {D} is not closed ?
Because, its immediate superset {A,D} also has the same support count 2.

Maximal & Closed Frequent Item set



All **Maximal Frequent Itemsets** are **Closed Frequent Itemsets** but all **Closed Frequent Itemsets** are **not Maximal Frequent Itemsets.**

Apriori Algorithm

Frequent Pattern Mining

- Based on the ***completeness* of patterns** to be mined.

Eg: closed frequent itemsets, and the maximal frequent itemsets, constrained frequent itemsets etc.

- 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.

Eg: single-dimensional association rule, multidimensional association rule.

$$buys(X, \text{"computer"}) \Rightarrow buys(X, \text{"antivirus\_software"})$$

$$age(X, \text{"30}\ldots\text{39"}) \wedge income(X, \text{"42K}\ldots\text{48K"}) \Rightarrow buys(X, \text{"high resolution TV"}).$$

- Based on the **types of values** handled in the rule.

Eg: Boolean association rule, quantitative association rule

- Based on the **kinds of rules** to be mined.

Eg: Association rules, correlation rules.

- Based on the **kinds of patterns** to be mined.

Eg: Sequential pattern mining, Structured pattern mining etc.

Frequent Itemset Mining Methods – Apriori Alg

It is for mining frequent itemsets for **boolean association rules**.
APRIORI Property: All nonempty subsets of a frequent itemset must also be frequent

Transactional data for an *AllElectronics* branch.

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

Two actions involved

1. **Join Step**
2. **Prune Step**

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

Generate C_2 candidates from L_1

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan D for count of each candidate

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count

L_2

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

Generate C_3 candidates from L_2

C_3

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

Scan D for count of each candidate

C_3

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

Compare candidate support count with minimum support count

L_3

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

Example -2

The Apriori Algorithm -- Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

Note: {1,2,3}{1,2,5} and {1,3,5} not in C_3

Steps of Apriori Algorithm

1. Generation of Candidate Item set C1.
2. Check for the required minimum support count of the transaction.
3. The set of frequent 1-itemset L1 is generated from C1 that satisfies the minimum support count(Pruning).
4. Discover the 2-frequent item set by L1*L1(Joining) and generate C2.
5. L2 is generated by pruning the records that do not satisfy the minimum support.
6. Discover the 3-frequent item set by L2*L2(Joining) and generate C3.
7. L3 is generated by pruning the records that do not satisfy the minimum support.
8. Discover the 4-frequent item set by L3*L3(Joining) and generate C4.
9. The Algorithm ends the frequent pattern mining if 4-frequent itemset is not available.

Apriori Algorithm

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:

```
L1 = find_frequent_1-itemsets(D);
for (k = 2; L(k-1) ≠ φ; k++) {
    Ck = apriori_gen(L(k-1));
    for each transaction t ∈ D { // scan D for counts
        Ct = subset(Ck, t); // get the subsets of t that are candidates
        for each candidate c ∈ Ct
            c.count++;
    }
    Lk = {c ∈ Ck | c.count ≥ min_sup}
}
return L = ∪k Lk;

procedure apriori_gen(L(k-1): frequent (k − 1)-itemsets)
    for each itemset l1 ∈ L(k-1)
        for each itemset l2 ∈ L(k-1)
            if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
                c = l1 ⋈ l2; // join step: generate candidates
                if has_infrequent_subset(c, L(k-1)) then
                    delete c; // prune step: remove unfruitful candidate
                else add c to Ck;
            }
    return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
        L(k-1): frequent (k − 1)-itemsets); // use prior knowledge
    for each (k − 1)-subset s of c
        if s ∉ L(k-1) then
            return TRUE;
    return FALSE;
```

- Find the frequent item sets in the following database with min support 50% & min confidence 50%.

Transaction id	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

- 50/100*4 = 2
- MIN SUP COUNT IS 2.

- Step1: find c1

Items	Support count
[A]	3
[B]	2
[C]	2
[D]	1
[E]	1
[F]	1

- Min support count is 2 so eliminate which are less than that.

- Step 2: compare the candidate support count with min support count so L1 will be

Items	Support
[A]	3
[B]	2
[C]	2

- Step 3: Generate candidate C2 from L1.

Items
[A,B]
[A,C]
[B,C]

- Step 4: Scan D for count of each candidate in C2 and find support.

Items	Support
[A,B]	1
[A,C]	2
[B,C]	1

- Step 5: Compare candidate C2 support count with min support count so L2 will be

Items	Support
[A,C]	2

- Step 6: so the data contains the frequent item [A,C].

Association Rule	Support	Confidence	Confidence %
A->C	2	2/3=0.66	66%
C->A	2	2/2=1	100%

Min Confidence - 50%

So final rules are

Rule 1: A -> C
Rule 2: C -> A

Generating Association Rules for Frequent Item sets

- Association rules can be generated as follows:

- 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 $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$, where min_conf is the minimum confidence threshold.

Generating Association Rules for Frequent Item sets

Generating association rules. Let's try an example based on the transactional data for *AllElectronics* shown in Table 5.1. Suppose the data contain the frequent itemset l = {I1, I2, I5}. What are the association rules that can be generated from l? The nonempty subsets of l are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, and {I5}. The resulting association rules are as shown below, each listed with its confidence:

$$
\begin{array}{ll}
I1 \wedge I2 \Rightarrow I5, & confidence = 2/4 = 50\% \\
I1 \wedge I5 \Rightarrow I2, & confidence = 2/2 = 100\% \\
I2 \wedge I5 \Rightarrow I1, & confidence = 2/2 = 100\% \\
I1 \Rightarrow I2 \wedge I5, & confidence = 2/6 = 33\% \\
I2 \Rightarrow I1 \wedge I5, & confidence = 2/7 = 29\% \\
I5 \Rightarrow I1 \wedge I2, & confidence = 2/2 = 100\%
\end{array}
$$

Minimum Confidence : 70%

Generating Association Rules for Frequent Item sets

- R1 I1^I2 -> I5

Confidence = SC(I1,I2,I5)/SC(I1,I2) = 2/4=50%. (Rejected)

- R2 I1^I5 -> I2

Confidence = SC(I1,I5,I2)/SC(I1,I5) = 2/2=100%. (Accepted)

- R3 I2^I5 -> I1

Confidence = SC(I2,I5,I1)/SC(I2,I5) = 2/2=100%. (Accepted)

- R4 I1->I2^I5

Confidence = SC(I1,I2,I5)/SC(I1) = 2/6=33%. (Rejected)

- R5 I2->I1^I5

Confidence = SC(I2,I1,I5)/SC(I2) = 2/7=29%. (Rejected)

- R6 I5->I1^I2

Confidence = SC(I5,I1,I2)/SC(I5) = 2/2=100%. (Accepted)

Problem - 1

- A database has **five** transactions. Let the Minimum Support & Confidence ,**min_sup=60%, min_confi = 100%.**
- Find the frequent itemsets and generate the association rules using **Apriori** algorithm.

TID	ITEMS
T1	{M,O,N,K,E,Y}
T2	{D,O,N,K,E,Y}
T3	{M,A,K,E}
T4	{M,U,C,K,Y}
T5	{C,O,O,K,I,E}

Problem - 2

- A database has **five** transactions. Let the Minimum Support & Confidence ,**min_sup=3, min_confi = 80%.**

TID	ITEMS
T1	{1,2,3,4,5,6}
T2	{7,2,3,4,5,6}
T3	{1,8,4,5}
T4	{1,9,0,4,6}
T5	{0,2,2,4,5}

- Find the frequent item sets and generate the association rules using **Apriori** algorithm.

Improving the Efficiency of Apriori

- **Transaction Reduction(reducing the number of transactions scanned in future iterations):** A transaction that does not contain any frequent k-itemsets cannot contain any frequent (k+1)-itemsets.

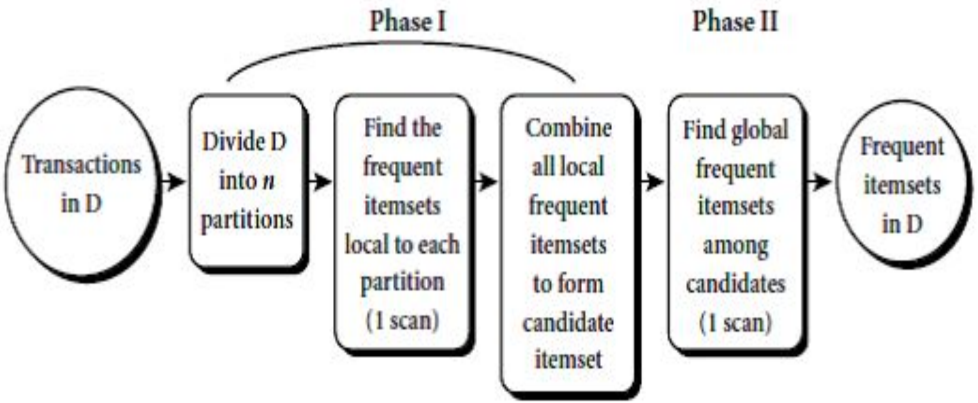
- **Partitioning(partitioning the data to find candidate itemsets):** Partitioning technique can be used that requires just two database scans to mine the frequent itemsets.
- In **Phase I**, the algorithm subdivides the transactions of D into n non overlapping partitions. If the minimum support threshold for transactions in D is min sup, then the minimum support count for a partition is

min sup X the number of transactions in that partition.

- All frequent itemsets within the partition are found. These are referred to as **local frequent itemsets**.

Improving the Efficiency of Apriori

- **Phase II**, *Any itemset that is potentially frequent with respect to D must occur as a frequent itemset in at least one of the partitions*. Therefore, all local frequent itemsets are candidate itemsets with respect to *D*.
- *The collection of frequent itemsets from all partitions forms the global candidate itemsets.*



Improving the Efficiency of Apriori

Sampling(mining on a subset of the given data):

- Pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D.

Dynamic itemset counting (adding candidate itemsets at different points during a scan):

- The database is partitioned into blocks marked by start points.
- new candidate itemsets can be added at any start point.

Hash Based techniques

- **Hash-based technique (hashing itemsets into corresponding buckets):**
- A hash-based technique can be used to reduce the size of the candidate *k*-itemsets.

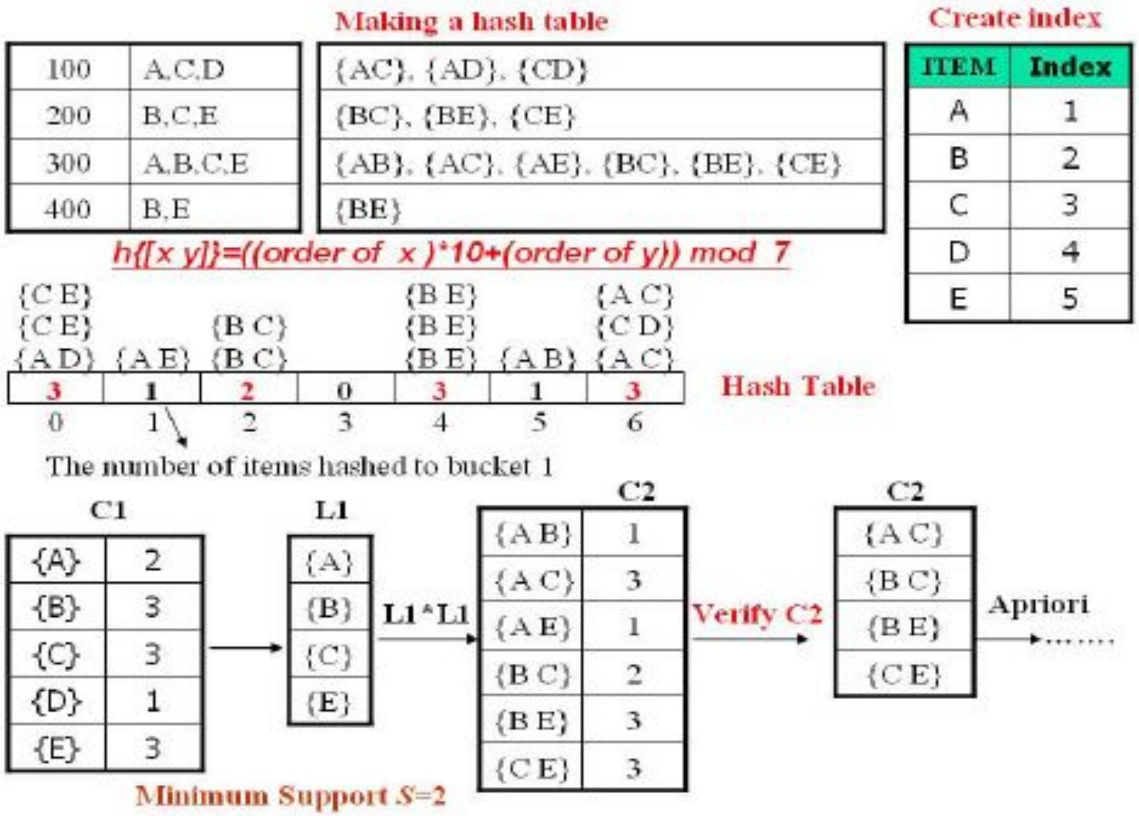
Create hash table H_2 using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y))\ mod\ 7$ →

H_2

bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3} {I1, I3}

Hash table, H_2, for candidate 2-itemsets: This hash table was generated by scanning the transactions of Table 5.1 while determining L_1 from C_1. If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in C_2.

Hash Based techniques



Frequent Pattern Growth Algorithm

Mining Frequent Item sets without Candidate Generation

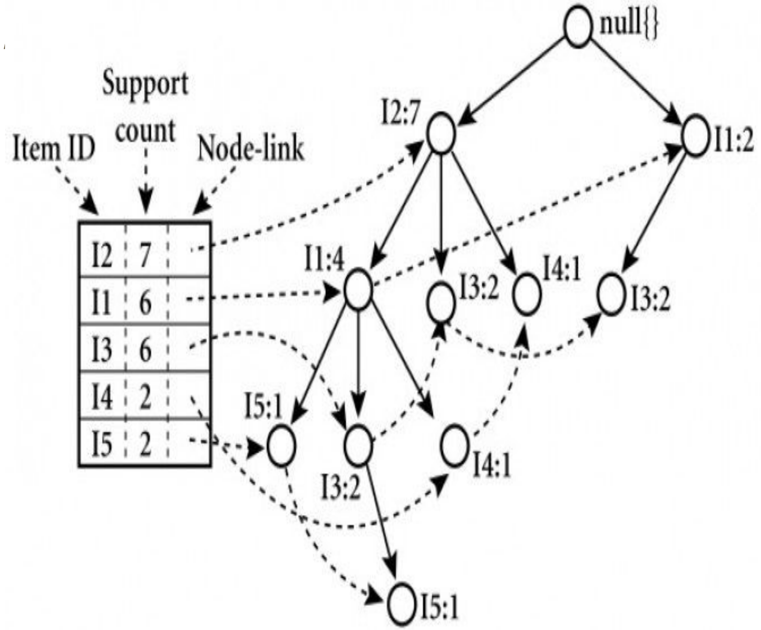
Disadvantages in **Apriori** Algorithm:

- *It may need to generate a huge number of candidate sets.*
- *It may need to repeatedly scan the database and check a large set of candidates by pattern match*

FP Growth Algorithm

Transactional data for an *AllElectronics* branch.

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



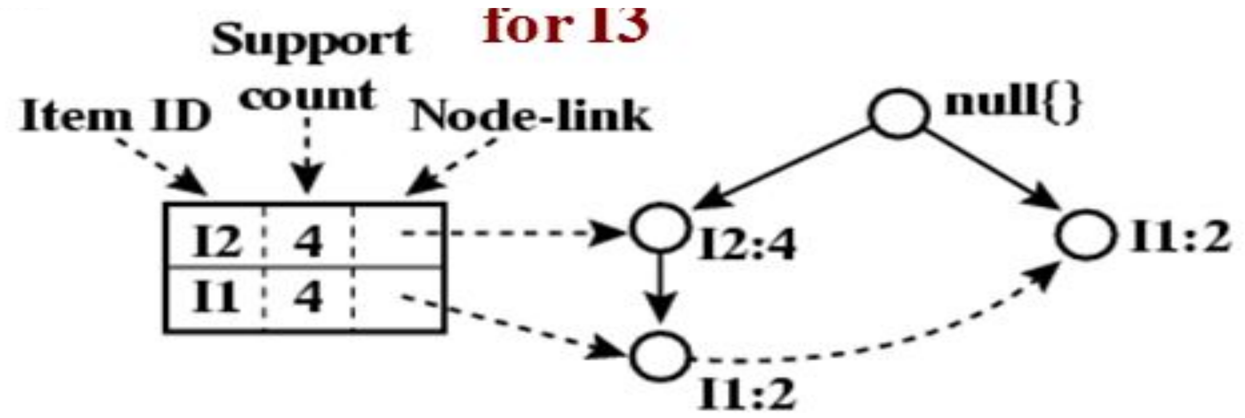
Mining Frequent Item sets without Candidate Generation

- we will start from the node that has the minimum support count ie.I5.
- We exclude the node with maximum support count ie.I2 for preparing the table.

Mining the FP-tree by creating conditional(sub-) pattern bases:

Item	Conditional Pattern Base	Conditional FP	Frequent Patterns Generated
I5	{{I2, I1:1}, {I2, I1, I3:1}}	{I2:2, I1:2}	{I2,I5:2},{I1,I5:2},{I2, I1,I5:2}
I4	{{I2, I1}, {I2:1}}	{I2:2}	{I2,I4:2}
I3	{{I2, I1:2},{I2:2}, {I1:2}}	{I2:4, I1:4}	{I2,I3:4},{I1,I3:4},{I2, I1,I3:2}
I1	{{I2:4}}	{I2:4}	{I2,I1:4}

The Conditional FP-Tree associated with the Conditional node I3.



FP Growth Algorithm

Algorithm: FP_growth. Mine frequent itemsets using an FP-tree by pattern fragment growth.

Input:

- D, a transaction database;
- *min_sup*, the minimum support count threshold.

Output: The complete set of frequent patterns.

Method:

1. The FP-tree is constructed in the following steps:

   (a) Scan the transaction database D once. Collect F, the set of frequent items, and their support counts. Sort F in support count descending order as L, the *list* of frequent items.

   (b) Create the root of an FP-tree, and label it as "null." For each transaction *Trans* in D do the following.

   Select and sort the frequent items in *Trans* according to the order of L. Let the sorted frequent item list in *Trans* be $[p|P]$, where p is the first element and P is the remaining list. Call insert_tree($[p|P]$, T), which is performed as follows. If T has a child N such that $N.item\text{-}name = p.item\text{-}name$, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same *item-name* via the node-link structure. If P is nonempty, call insert_tree(P, N) recursively.

2. The FP-tree is mined by calling FP_growth(*FP_tree*, *null*), which is implemented as follows.

procedure FP_growth($Tree$, α)

if $Tree$ contains a single path P **then**

    for each combination (denoted as β) of the nodes in the path P

        generate pattern $\beta \cup \alpha$ with $support\_count = minimum\ support\ count\ of\ nodes\ in\ \beta$;

else for each a_i in the header of $Tree$ {

    generate pattern $\beta = a_i \cup \alpha$ with $support\_count = a_i.support\_count$;

    construct β's conditional pattern base and then β's conditional FP_tree $Tree_\beta$;

    if $Tree_\beta \neq \emptyset$ **then**

        call FP_growth($Tree_\beta$, β); }

FP Growth Algorithm Vs Apriori Algorithm

FP Growth Algorithm	Apriori Algorithm
1. FP growth algorithm is faster than Apriori algorithm.	It is slower than FP growth algorithm .
2. It is an array based algorithm.	It is a tree based algorithm
3. It required only 2 database scan	It requires multiple database scan to generate a candidate set.
4.It uses depth-first search	It uses breath-first search.
5. Less accurate	More accurate

FP GROWTH ALGORITHM Vs APRIORI ALGORITHM

File	Apriori	FP-Growth
Simple Market Basket test file	3.66 s	3.03 s
"Real" test file (1 Mb)	8.87 s	3.25 s
"Real" test file (20 Mb)	34 m	5.07 s
Whole "real" test file (86 Mb)	4+ hours (Never finished, crashed)	8.82 s

Problems 1 – FP Growth Tree

- A database has **five** transactions. Let the Minimum Support **min_sup=60%.**
- Find the frequent itemsets using FP growth Algorithm.

TID	ITEMS
T1	{M,O,N,K,E,Y}
T2	{D,O,N,K,E,Y}
T3	{M,A,K,E}
T4	{M,U,C,K,Y}
T5	{C,O,O,K,I,E}

Problems 2 – FP Growth Tree

- A database has **Eight** transactions. Let the Minimum Support, **min_sup=30%.**

TID	ITEMS
1	{E,A,D,B}
2	{D,A,C,E,B}
3	{C,A,B.E}
4	{B,A,D}
5	{D}
6	{D,B}
7	{A,D,E}
8	{B,C}

- Find the frequent item sets using **FP growth** Algorithm.

Mining Frequent Item sets Using Vertical Data Format

Horizontal Data Format will be converted to Vertical Data Format

Transactional data for an *AllElectronics* branch.

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

The vertical data format of the transaction data set *D* of Table 5.1.

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}

Mining Frequent Item sets Using Vertical Data Format

The 2-itemsets in vertical data 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}

The 3-itemsets in vertical data format.

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

Mining Closed Frequent Item sets

- It is a frequent itemset that is both closed and its support is greater than or equal to minsup.
- An itemset is closed in a data set if there exists no superset that has the same support count as this original itemset.

- Frequent itemset mining may generate a huge number of frequent itemsets, when the ***min sup*** **threshold** is set **low** or when there exist **long patterns** in the data set.

Mining Closed Frequent Item sets

“How can we mine closed frequent itemsets?”

- First mine the complete set of frequent itemsets.
- Then remove every frequent itemset that is a proper subset of, and carries the same support as, an existing frequent itemset.

- To search for closed frequent itemsets directly during the mining process.
- This requires us to prune the search space as soon as we can identify the case of closed itemsets during mining.

Pruning strategies

- **Item merging:** *If every transaction containing a frequent item set X also contains an item set Y but not having any proper superset of Y,*
- *then X* U*Y forms a frequent closed item set and there is no need to search for any item set containing X but no Y.*

- Projected database for {I5: 2} is {{I2, I1}, {I2,I1,I3}}. Each transaction contains item-set {I2, I1} but no proper superset of {I2, I1}. So this can be merged with {I5} to give {I5, I2, I1:2}
- There is no need to mine for closed item-sets that contain I5 but not {I2, I1}

Pruning strategies

- ***Sub-item set pruning:*** *If a frequent item set X is a proper subset of an already found frequent closed itemset Y*
- *and support count(X) = support count(Y), then X and all of X's descendants in the set enumeration tree cannot be frequent closed item sets and thus can be pruned.*

  - { <a1, a2, …,a100>, <a1, a2,…,a50>} min_sup = 2
  - Projection on a1 gives {a1, a2,…,a50 : 2} based on Itemset merging
  - Support {a2} = support ({a1, a2,..a50}) = 2 and a2 is a proper subset - no need to examine a2 and its projections

Pruning strategies

Item skipping: *In the depth-first mining of closed itemsets, at each level, there will be a prefix itemset X associated with a header table and a projected database.*

- *If a local frequent item p has the same support in several header tables at different levels, we can safely prune p from the header tables at higher levels*.

  – For example, a transaction database: $\{\langle a_1, a_2, \cdots, a_{100}\rangle, \langle a_1, a_2, \cdots, a_{50}\rangle\}$, min_sup = 2. Because a_2 in a_1's projected database has the same support as a_2 in the global header table, a_2 can be pruned from the global header table.

Pruning strategies

- Important **optimization** is to perform efficient **checking**

Perform **two** kinds of **closure checking**:

- ***superset checking:*** checks if this new frequent itemset is a superset of some already found closed itemsets with the same support.
- ***subset checking:*** checks whether the newly found itemset is a subset of an already found closed itemset with the same support.
- For **efficient subset checking**, we can use the following property:
- *If the current itemset Sc can be subsumed by another already found closed itemset Sa, then*
  (1) Sc and Sa have the same support.
  (2) the length of Sc is smaller than that of Sa.
  (3) all of the items in Sc are contained in Sa.

Which Patterns Are Interesting?—Pattern Evaluation Method

- Most association rule mining algorithms employ a support-confidence framework.
- Many interesting rules can be found using low support thresholds.
- **Strong Rules Are Not Necessarily Interesting.**
- Whether or not a rule is interesting can be assessed either subjectively or objectively.
- only the user can judge if a given rule is interesting, and this judgment, being **subjective**, may differ from one user to another.
- **objective** interestingness measures, based on the statistics “behind” the data.

Association Mining to Correlation Analysis

A **misleading "strong" association rule.**

- 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 **6,000** of the customer transactions included **computer games**, while **7,500** included **videos**, and **4,000** included both **computer games and videos**.
- minimum support - 30% minimum confidence - 60%.
- Support value of computer games: 4000/10000 = 40%
- Confidence value of " " " " : 4000/6000 = 66%

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

Association Mining to Correlation Analysis

- 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.

From Association Analysis to Correlation Analysis

- The support and confidence measures are insufficient at filtering out uninteresting association rules.
- This leads to *correlation rules* of the form
  ***A=>B* [*support, confidence*. *correlation*].**
- A correlation rule is measured not only by its support and confidence but also by the correlation between item sets *A* and *B*.

Correlation Measures

- **Lift** is a simple correlation measure.
- The occurrence of item set A is independent of the occurrence of itemset B if $P(A \cup B) = P(A)P(B)$.
- otherwise, iter ... rrelated as events.

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}.$$

- Lift(A,B)<1 – A & B are negatively correlated.
- Lift(A,B)>1 – A & B are positively correlated.
- Lift(A,B)=1 – A & B are not correlated, they are independent.
- It assesses the degree to which the occurrence of one "lifts" the occurrence of the other.

equivalent to $P(B|A)/P(B)$, or $conf(A \Rightarrow B)/sup(B)$,

Correlation analysis using lift

A 2 × 2 contingency table summarizing the transactions with respect to game and video purchases.

	game	$\overline{game}$	Σ_{row}
video	4,000	3,500	7,500
$\overline{video}$	2,000	500	2,500
Σ_{col}	6,000	4,000	10,000

$$P(\{game\}) = 0.60 \qquad P(\{video\}) = 0.75$$

$$P(\{game, video\}) = 0.40$$

$$P(\{game, video\}) / (P(\{game\}) \times P(\{video\})) = 0.40 / (0.60 \times 0.75) = 0.89$$

0.89<1 so game and video are negatively correlated.

Correlation analysis using Chi square

The above contingency table, now shown with the expected values.

	$game$	$\overline{game}$	Σ_{row}
video	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{video}$	2,000 (1,500)	500 (1,000)	2,500
Σ_{col}	6,000	4,000	10,000

Correlation analysis using Chi square

Correlation analysis using χ^2

$$\chi^2 = \Sigma\frac{(observed - expected)^2}{expected} = \frac{(4,000 - 4,500)^2}{4,500} + \frac{(3,500 - 3,000)^2}{3,000} + \frac{(2,000 - 1,500)^2}{1,500} + \frac{(500 - 1,000)^2}{1,000} = 555.6.$$

Because the χ^2 value is greater than one, and the observed value of the slot (*game, video*) = 4,000, which is less than the expected value 4,500, *buying game* and *buying video* are *negatively correlated.*

General Rules:

$$\chi^2 = 0, independent$$

$$\chi^2 > 0 \text{ Correlated either + ve or } - \text{Ve. Needs additional tests}$$

Given an itemset $X = \{i_1, i_2, \ldots, i_k\}$, the **all_confidence** of X is defined as

$$all\_conf(X) = \frac{sup(X)}{max\_item\_sup(X)} = \frac{sup(X)}{max\{sup(i_j) | \forall i_j \in X\}},$$

Given two itemsets A and B, the **cosine** measure of A and B is defined as

$$cosine(A, B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}}.$$

Thank You