Unit - II

Mining Frequent Patterns, Associations, Correlations

(Contents: Text book 2 - Chapter 6)

Dr. R. Elakkiya
AP-III, SOC
SASTRA Deemed University

Mining Frequent Patterns, Associations, and Correlations

- In this topic,
  - Basic concepts
    - Market Basket Analysis: A Motivating Example
    - Frequent Itemsets, Closed Itemsets, and Association Rules
  - Frequent Itemset Mining Methods:
    - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
    - A Pattern-Growth Approach for Mining Frequent Itemsets
    - Mining Frequent Itemsets Using Vertical Data Format
    - Mining Closed and Max Patterns
  - Which Patterns Are Interesting?—Pattern Evaluation Methods

Mining Frequent Patterns, Associations, and Correlations

- **Imagine that you are** a sales manager at *AllElectronics*, and you are talking to a customer who recently bought a PC and a digital camera from the store.
- What should you recommend to her next?
- Information about which products are frequently purchased by your customers following their purchases of a PC and a digital camera in sequence would be very helpful in making your recommendation.
- Frequent patterns and association rules are the knowledge that you want to mine in such a scenario.

Mining Frequent Patterns, Associations, and Correlations

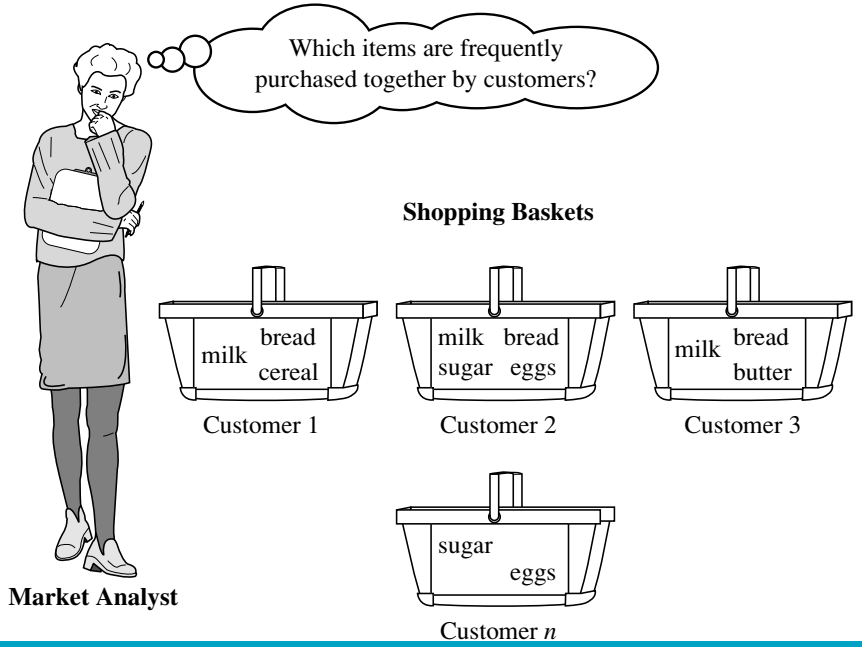
- **Frequent patterns** are patterns (e.g., itemsets, subsequences, or substructures) that appear frequently in a data set.
- For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set is a *frequent itemset*.
- A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (*frequent*) *sequential pattern*.
- A *substructure* can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (*frequent*) *structured pattern*.
- Finding frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data.
- Moreover, it helps in data classification, clustering, and other data mining tasks.

6.1 Basic Concepts

- Frequent pattern mining searches for recurring relationships in a given data set.
- Basic concepts of frequent pattern mining for the discovery of interesting associations and correlations between itemsets in transactional and relational databases.
  - Example of market basket analysis, the earliest form of frequent pattern mining for association rules.
  - Basic concepts of mining frequent patterns and associations

6.1.1 Market Basket Analysis: A Motivating Example

- Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets.
- With massive amounts of data continuously being collected and stored, many industries are becoming interested in mining such patterns from their databases.
- Help in many business decision-making processes such as catalog design, cross-marketing, and customer shopping behavior analysis.
- A typical example of frequent itemset mining is **market basket analysis** - customer buying habits by finding associations between the different items that customers place in their "shopping baskets"



Market Basket Analysis: A Motivating Example

- Suppose, as manager of an *AllElectronics* branch, you would like to learn more about the buying habits of your customers.
- Specifically, you wonder, *"Which groups or sets of items are customers likely to purchase on a given trip to the store?"*
- Each item has a Boolean variable representing the presence or absence of that item.
- Each basket can then be represented by a Boolean vector of values assigned to these variables.
- The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently *associated* or purchased together. These patterns can be represented in the form of **association rules**.
- Ex: customers who purchase computers also tend to buy antivirus software at the same time is represented in the following association rule:

$$computer \Rightarrow antivirus\ software\ [support = 2\%, confidence = 60\%]$$

Market Basket Analysis: A Motivating Example

- Rule **support** and **confidence** are two measures of rule interestingness.
- support (2%) - 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
- confidence (60%) - 60% of the customers who purchased a computer also bought the software.
- Typically, association rules are considered interesting if they satisfy both a **minimum support threshold** and a **minimum confidence threshold**.
- These thresholds can be a set by users or domain experts.
- Rules that satisfy both minsup and minconf are called strong rules
- Additional analysis can be performed to discover interesting statistical correlations between associated items.

6.1.2 Frequent Itemsets, Closed Itemsets, and Association Rules

- Let I = {$I1, I2,..., Im$} be an itemset.
- Let D, the task-relevant data, be a set of database transactions where each transaction T is a nonempty itemset such that $T \subseteq$ I.
- Let A be a set of items. A transaction T is said to contain A if $A \subseteq T$.
- An association rule is an implication of the form $A \Rightarrow B$, where $A \subset$ I, $B \subset$ I, $A \neq \emptyset$, $B \neq \emptyset$, and $A \cap B = \varphi$.
- The rule $A \Rightarrow B$ - **support** s, where s is the percentage of transactions in D that contain $A \cup B$. This is taken to be the probability, $P(A \cup B)$
- The rule $A \Rightarrow B$ - **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, $P(B|A)$.

$$support(A \Rightarrow B) = P(A \cup B)$$

$$confidence\ (A \Rightarrow B) = P(B|A)$$

Frequent Itemsets, Closed Itemsets, and Association Rules

- A set of items is referred to as an **itemset**.
- An itemset that contains k items is a ***k*-itemset**. Ex: The set {*computer, antivirus software*} is a 2-itemset.
- The **occurrence frequency of an itemset** is the number of transactions that contain the itemset. (**frequency**, **support count**, or **count** of the itemset).

$$support(A \Rightarrow B) = P(A \cup B)$$

- The above itemset support is sometimes referred to as ***relative support***, whereas the occurrence frequency is called the **absolute support**.
- If the relative support of an itemset I satisfies a prespecified **minimum support threshold** (i.e., the absolute support of I satisfies the corresponding **minimum support count threshold**), then I is a **frequent** itemset.
- The set of frequent k-itemsets is commonly denoted by L_k.

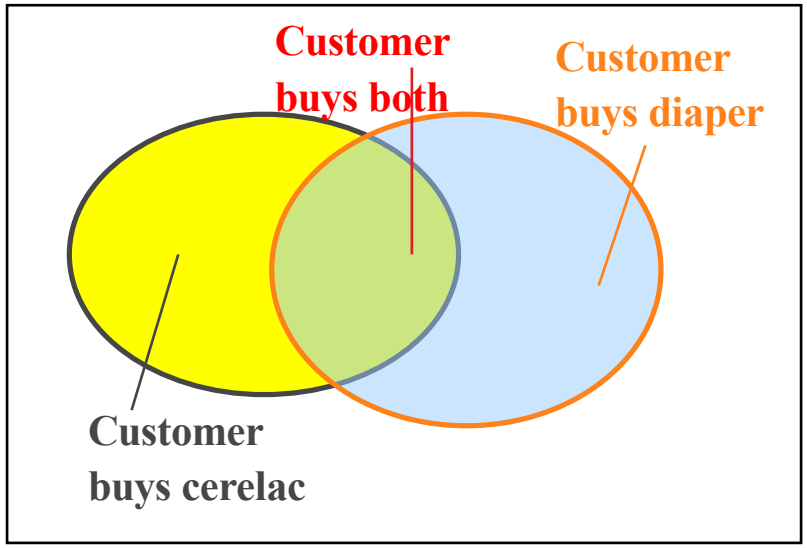
$$\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\text{support}(A \cup B)}{\text{Support}(A)} = \frac{\text{support\_count}(A \cup B)}{\text{Support\_Count}(A)}$$

Frequent Itemsets, Closed Itemsets, and Association Rules

- In general, association rule mining can be viewed as a two-step process:
  - **Find all frequent itemsets:** By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min sup*.
  - **Generate strong association rules from the frequent itemsets:** By definition, these rules must satisfy minimum support and minimum confidence.

Example

Tid	Items bought
10	cerelac, Nuts, Diaper
20	cerelac, Coffee, Diaper
30	cerelac, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- 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 minsup = 50%, minconf = 50%

Ex: cerelac_count = 3, support = 3/5 =0.6 (60%)

Freq. Pat.: cerelac:3, Nuts:3, Diaper:4, Eggs:3, {cerelac, Diaper}:3

Confidence (Cer =>Dia) = sup(Cer U Dia)/Sup (Cer) = (3/5)/(3/5) = 1(100%)
Confidence (Dia => Cer) = sup(Dia U Cer)/Sup (Dia) = (3/5)/(4/5) = 0.75 (75%)

- Association rules: (many more!)
  - cerelac → *Diaper* (60%, 100%)
  - *Diaper* → cerelac (60%, 75%)

Closed Patterns and Max-Patterns

- A major challenge in mining frequent itemsets from a large data set is the fact that such mining often generates a huge number of itemsets satisfying the minimum support (*min sup*) threshold, especially when *min sup* is set low.
- This is because if an itemset is frequent, each of its subsets is frequent as well.
- A long itemset will contain a combinatorial number of shorter, frequent sub-itemsets.

Closed Patterns and Max-Patterns

- How many frequent itemsets does the following TDB_1 contain?
- Assuming(absolute) minsup=1
  - Let's have a try
  - TDB_1 $T_1: \{a_1, \dots, a_{50}\}$; $T_2: \{a_1, \dots, a_{100}\}$

    1-itemsets: $(a_1): 2, (a_2): 2, \dots, (a_{50}): 2, (a_{51}): 1, \dots, (a_{100}): 1$

    2-itemsets: $(a_1, a_2): 2, \dots, (a_1, a_{50}): 2, (a_1, a_{51}): 1 \ \dots, \dots, (a_{99}, a_{100}): 1$

    …, …, …,

    99-itemsets: $\{a_1, a_2, \dots, a_{99}\}: 1, \dots, \{a_2, a_3, \dots, a_{100}\}: 1$

    100-itemset: $\{a_1, a_2, \dots, a_{100}\}: 1$
- In total: $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1$

  $\approx 1.27\ X\ 1030\ Subpatterns$

A too huge set for Any computer to compute or store!

Closed Patterns and Max-Patterns

- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is **closed** in a data set D if there exists no proper super-itemset Y such that Y has the same support count as X in D.
- TDB_1 = {<a_1, …, a_{100}>, < a_1, …, a_{50}>} and Min_sup = 1.
- How many closed patterns does TDB_1 contain?
  - Closed itemset C = {{$a1$, $a2$,…, $a100$} : 1; {$a1$, $a2$,…, $a50$} : 2}
    - {$a2$, $a45$: 2} since {$a2$, $a45$} is a sub-itemset of the itemset {$a1$, $a2$,…, $a50$: 2};
    - {$a8$, $a55$: 1} since {$a8$, $a55$} is not a sub-itemset of the previous itemset but of the itemset {$a1$, $a2$,…, $a100$: 1}.
- Closed pattern is a lossless compression of frequent patterns.
  - Reduces the # of patterns but does lose the support information!
- An itemset X is a **closed frequent itemset** in set D if X is both closed and frequent in D.

Closed Patterns and Max-Patterns

- An itemset X is a **maximal frequent itemset** (or **max-itemset**) in a data set D if X is frequent, and there exists no super-itemset Y such that $X \subset Y$ and Y is frequent in D.
  - $<a_1, \ldots, a_{100}>: 1$
    - From the maximal frequent itemset, we can only assert that both itemsets ($\{a2, a45\}$ and $\{a8, a55\}$) are frequent, but we cannot assert their actual support counts.
    - Max-pattern is a lossy compression!
      - We only Know $(a_1, \ldots, a_{40})$ is frequent.
      - But we do not know the real support of $(a_1, \ldots, a_{40})$, …,any more!

Closed Patterns and Max-Patterns

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

- When we have many items, the total number of itemsets grow exponentially to the number of items in the database: |itemsets| = 2^n. This immediately causes both space and time complexity problems in finding frequent patterns.
- For example, the table above contains five transaction records and six items. This small database can have $2^6 = 64$ possible itemsets.
- To deal with this problem, we now define a series of sets.

Sub-pattern (proper subset) & Super-pattern (proper superset)

- To find efficient algorithms for mining frequent patterns, we will start with two basic concepts of patterns.
- A pattern is an itemset, therefore the order of the items are not important.
- What will be useful is the properties of subsets or supersets of frequent patterns.
- Let us start with subsets: any proper subset of a pattern, a sub-pattern.
- Ex: If a pattern X contains, 3 items: X={a, b, c}, any proper subsets, such as sets containing a, or b, or c, or ab, or ac, or bc, are all sub patterns.
- Another concept is of course super-pattern. A super pattern of a pattern is a proper super set of the pattern.
- Here are some examples of the pattern containing the 3 items: a, b, c
- abcd, abce, abcde, …

Close Pattern & Max Pattern

- A **closed pattern** is a frequent pattern. So it meets the minimum support criteria.
- In addition to that, all super-patterns of a closed pattern are less frequent than the closed pattern.
- Ex: 3 items: a, b, c. Suppose, the minimum support count is 2.
- Suppose a pattern ab has support count of 2 and a pattern abc has support count of 2. Is the pattern ab is a closed pattern?
- No. Pattern ab is a frequent pattern, but it has a super-pattern that is NOT less frequent than ab.
- Ex: 3 items: x, y, z.
- suppose a pattern xy has support count of 3 and a pattern xyz has support count of 2. Is the pattern xy is a closed pattern?
- Pattern xy is a frequent pattern and also the only super-pattern xyz is less frequent than xy.
- Therefore, xy is a closed pattern.

Close Pattern & Max Pattern

- A **max pattern** is a frequent pattern. So it also meets the minimum support criteria like closed pattern
- In addition, but unlike closed pattern, all super-patterns of a max pattern are NOT frequent patterns.
- Ex: 3 items: a, b, c. Suppose, the minimum support count is 2.
- Suppose a pattern ab has support count of 3 and a pattern abc has support count of 2. Is the pattern ab is a max pattern?
- Pattern ab is a frequent pattern, but it has a super-pattern that is a frequent pattern as well. So, pattern ab is NOT a max pattern.
- Ex: 3 items: x, y, z. Suppose, the minimum support count is 2.
- Suppose a pattern xy has support count of 3 and a pattern xyz has support count of 1. Is the pattern xy is a max pattern?
- Pattern xy is a frequent pattern and also the only super-pattern xyz is NOT a frequent pattern. Therefore, xy is a max pattern.

6.2 Frequent Itemset Mining Methods

- In this topic, methods for mining the simplest form of frequent patterns
  - Apriori, the basic algorithm for finding frequent itemsets
  - Generate strong association rules from frequent itemsets.
  - several variations to the Apriori algorithm for improved efficiency and scalability.
  - Frequent Pattern-growth methods for mining frequent itemsets
  - Methods for mining frequent itemsets from vertical data format.

6.2.1 Apriori Algorithm: Finding Frequent Itemsets by Confined Candidate Generation

- Apriori is a seminal algorithm for mining frequent itemsets for Boolean association rules
- 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 by 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.
- To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the **Apriori property** is used to reduce the search space.

Apriori Algorithm:

- **Apriori property:** *All nonempty subsets of a frequent itemset must also be frequent.*

- if an itemset I does not satisfy the minimum support threshold, *min sup*, then I is not frequent, that is, $P(I) < min\ sup$.
- If an item A is added to the itemset I, then the resulting itemset (i.e., $I \cup A$) cannot occur more frequently than I . Therefore, $I \cup A$ is not frequent either, that is, $P(I \cup A) < min\ sup$.
- This property belongs to a special category of properties called **antimonotonicity** in the sense that *if a set cannot pass a test, all of its supersets will fail the same test as well*. It is called *antimonotonicity* because the property is monotonic in the context of failing a test.

Apriori Algorithm:

- "How is the Apriori property used in the algorithm?"
- To understand this, let us look at how L_{k-1} is used to find L_k for $k \geq 2$.
- A two-step process
  - **Join:** 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.
  - **Prune:** 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

Apriori Algorithm:

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - 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

The Apriori Algorithm (Pseudo-Code)

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)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ φ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // scan D for counts
(5)           Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck|c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;
```

The Apriori Algorithm (Pseudo-Code)

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

(1) **for each** itemset $l_1 \in L_{k-1}$
(2) **for each** itemset $l_2 \in L_{k-1}$
(3) **if** $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2])$
$\wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ **then** {
(4) $c = l_1 \bowtie l_2$; // join step: generate candidates
(5) **if** 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 has_infrequent_subset(c: candidate k-itemset;
L_{k-1}: frequent $(k-1)$-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;

Apriori: Example

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

Example

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

Example

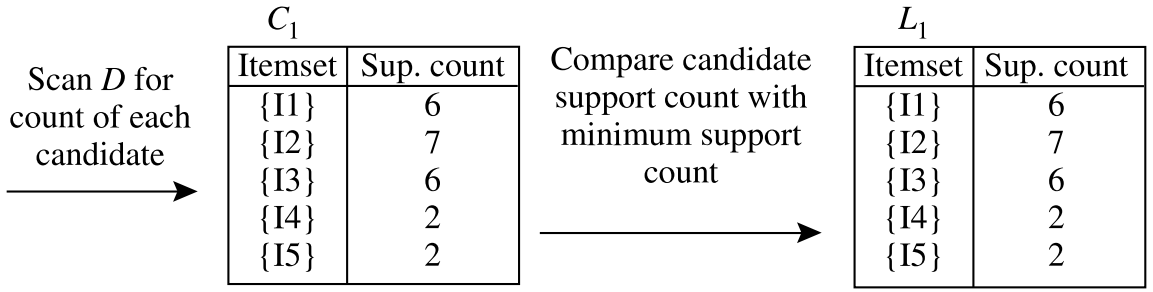
(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$\bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}.$

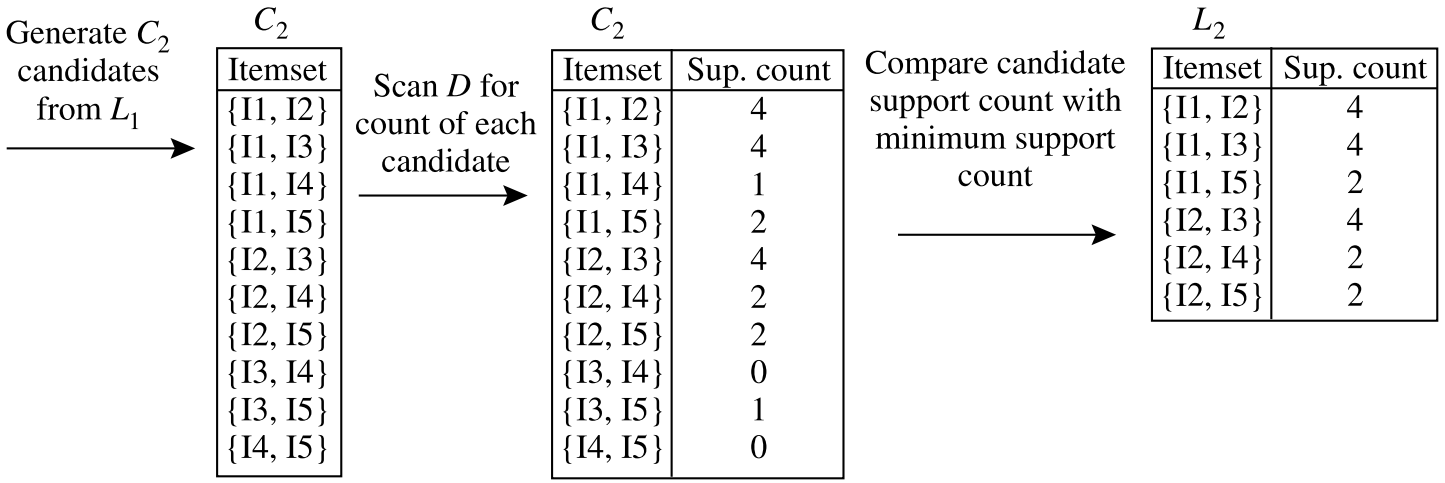
(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

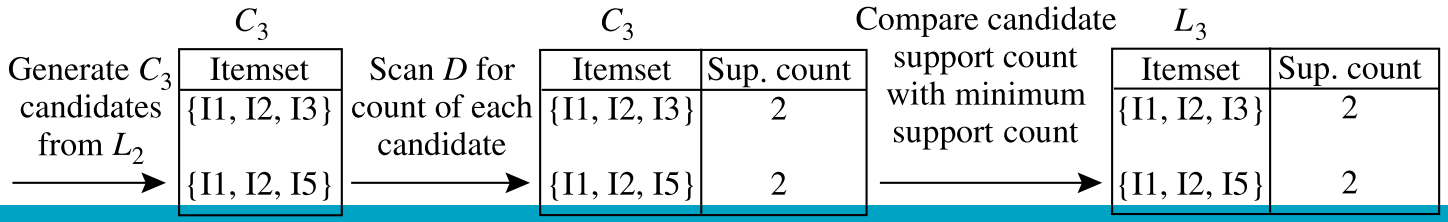
- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Apriori: Example







6.2.2 Generating Association Rules from Frequent Itemsets

- The data contain frequent itemset $X = \{I1, I2, I5\}$.
- What are the association rules that can be generated from X?
- The nonempty subsets of X are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, and {I5}.
- The resulting association rules are as shown below, each listed with its confidence:
- $\{I1, I2\} \Rightarrow I5$, *confidence* = 2/4 = 50%
- $\{I1, I5\} \Rightarrow I2$, *confidence* = 2/2 = 100%
- $\{I2, I5\} \Rightarrow I1$, *confidence* = 2/2 = 100%
- $I1 \Rightarrow \{I2, I5\}$, *confidence* = 2/6 = 33%
- $I2 \Rightarrow \{I1, I5\}$, *confidence* = 2/7 = 29%
- $I5 \Rightarrow \{I1, I2\}$, *confidence* = 2/2 = 100%

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

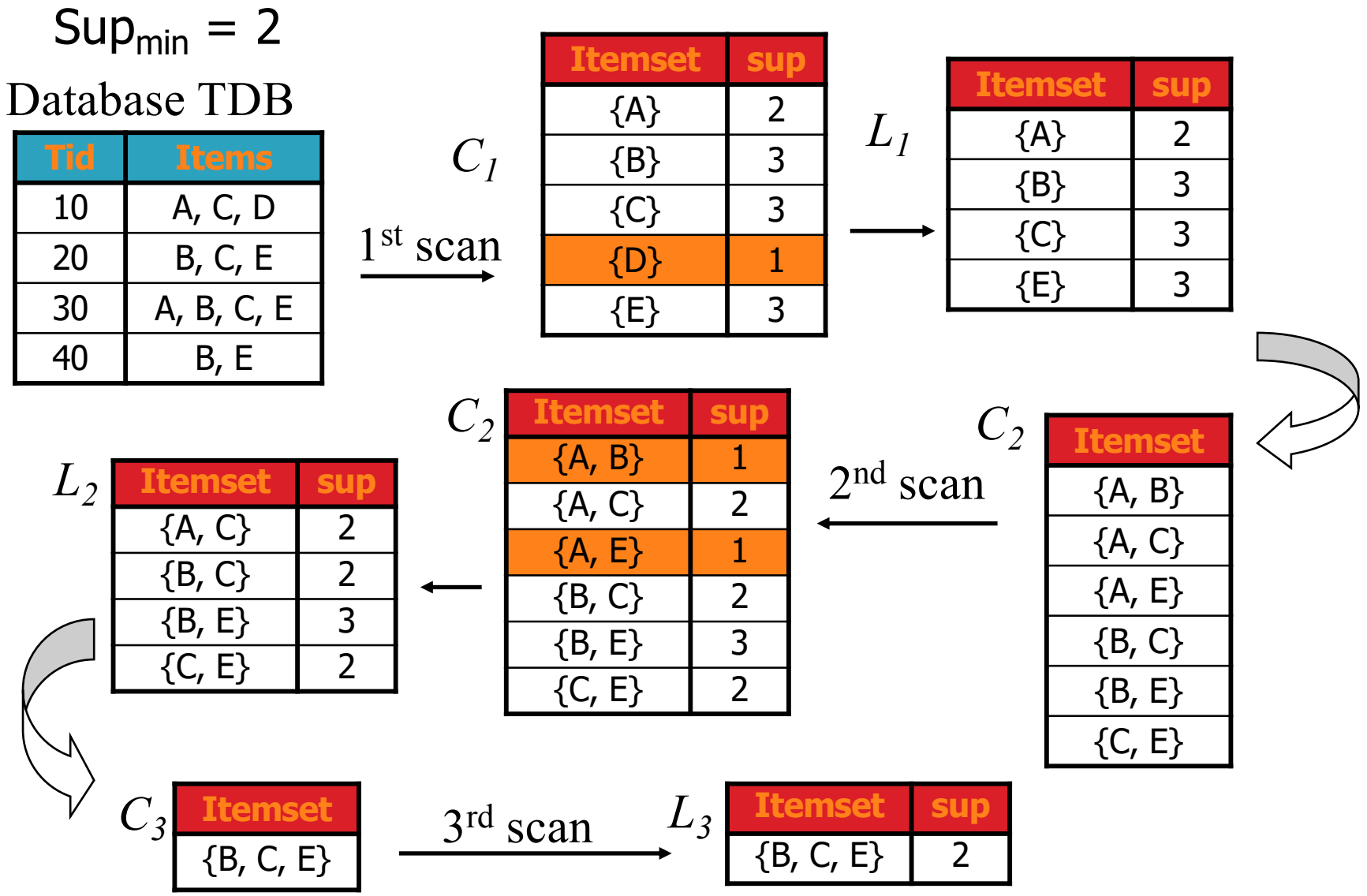
Another Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

Another Example



6.2.3 Improving the Efficiency of Apriori

- Hash-Based Technique:
- A hash-based technique can be used to reduce the size of the candidate *k*-itemsets, *Ck* , for $k > 1$.

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

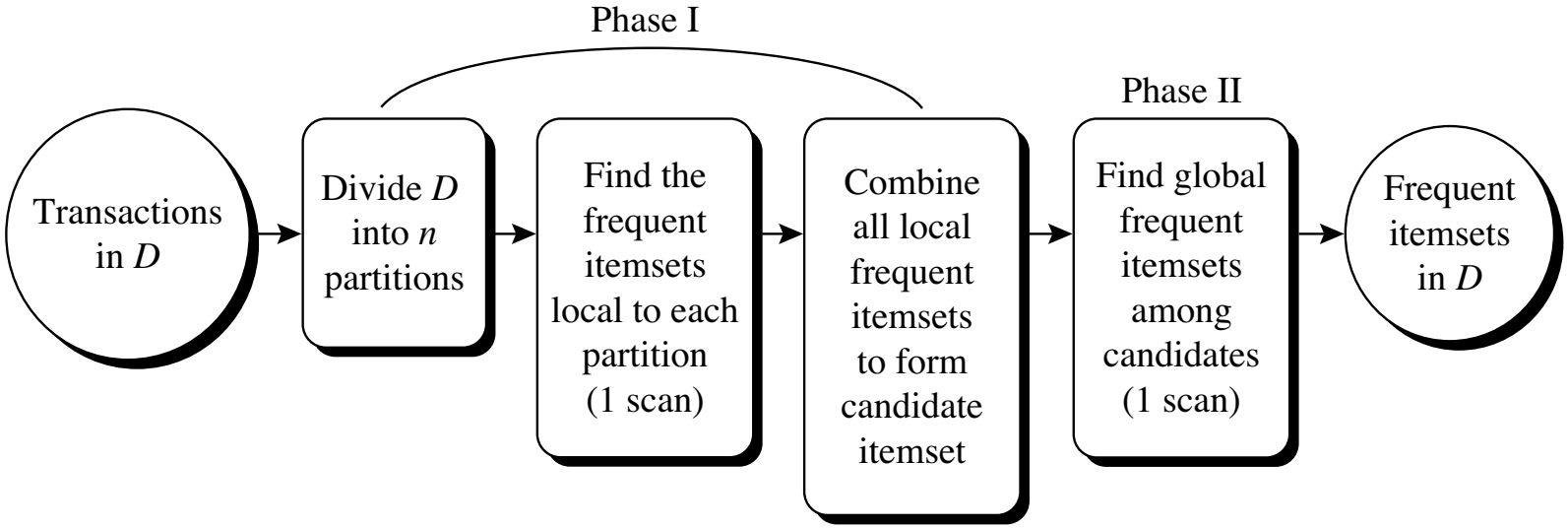
H_2

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}

- Ex: Generate L_1 – 1-frequent itemset and construct hash table
- A 2-itemset with a corresponding bucket count in the hash table that is below the support threshold cannot be frequent and thus should be removed from the candidate set.
- Such a hash-based technique may substantially reduce the number of candidate *k*-itemsets examined (especially when $k = 2$).

Improving the Efficiency of Apriori

- **Transaction reduction** (reducing the number of transactions scanned in future iteraions):
  - 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):
  - A partitioning technique can be used that requires just two database scans to mine the frequent itemsets.



Improving the Efficiency of Apriori

- **Two Phases**:
  - In phase I, the algorithm divides the transactions of D into n nonoverlapping partitions.
  - If the minimum relative support threshold for transactions in D is *min sup*, then the minimum support count for a partition is *min_sup × the number of transactions in that partition*.
  - For each partition, all the *local frequent itemsets* (i.e., the itemsets frequent within the partition) are found.
  - The collection of frequent itemsets from all partitions forms the *global candidate itemsets* with respect to D
  - In phase II, a second scan of D is conducted in which the actual support of each candidate is assessed to determine the global frequent itemsets.

Improving the Efficiency of Apriori

- **Sampling** (mining on a subset of the given data):
  - The basic idea of the sampling approach is to pick a random sample *S* of the given data *D*, and then search for frequent itemsets in *S* instead of *D*.
  - In this way, we trade off some degree of accuracy against efficiency.
- **Dynamic itemset counting** (adding candidate itemsets at different points during a scan):
  - A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points.
  - In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan.

6.2.4 A Pattern-Growth Approach for Mining Frequent Itemsets

- In many cases the Apriori candidate generate-and-test method significantly reduces the size of candidate sets, leading to good performance gain.
- However, it can suffer from two nontrivial costs:
  - *It may still need to generate a huge number of candidate sets.*
    - For example, if there are 104 frequent 1-itemsets, the Apriori algorithm will need to generate more than 107 candidate 2-itemsets.
  - *It may need to repeatedly scan the whole database and check a large set of candidates by pattern matching.*
  - It is costly to go over each transaction in the database to determine the support of the candidate itemsets.

A Pattern-Growth Approach for Mining Frequent Itemsets

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)
  - Depth-first search
  - Avoid explicit candidate generation
- Major philosophy: Grow long patterns from short ones using local frequent items only
  - "abc" is a frequent pattern
  - Get all transactions having "abc", i.e., project DB on abc: DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

A Pattern-Growth Approach for Mining Frequent Itemsets

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-name* = *p.item-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.

A Pattern-Growth Approach for Mining Frequent Itemsets

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

procedure FP_growth(*Tree*, α)
(1) **if** *Tree* contains a single path *P* **then**
(2) **for each** combination (denoted as β) of the nodes in the path *P*
(3) generate pattern $\beta \cup \alpha$ with *support_count* = *minimum support count of nodes in* β;
(4) **else for each** a_i in the header of *Tree* {
(5) generate pattern $\beta = a_i \cup \alpha$ with *support_count* = a_i.*support_count*;
(6) construct β's conditional pattern base and then β's conditional FP_tree $Tree_\beta$;
(7) **if** $Tree_\beta \neq \emptyset$ **then**
(8) call FP_growth($Tree_\beta$, β); }

Example

Transaction ID	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{C,K,M,U,Y}
T5	{C,E,I,K,O,O}

Support_count of itemsets

Transaction ID	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{C,K,M,U,Y}
T5	{C,E,I,K,O,O}

Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

Frequent Patterns are

L = {K : 5, E : 4, M : 3, O : 3, Y : 3}

Ordered-Item set

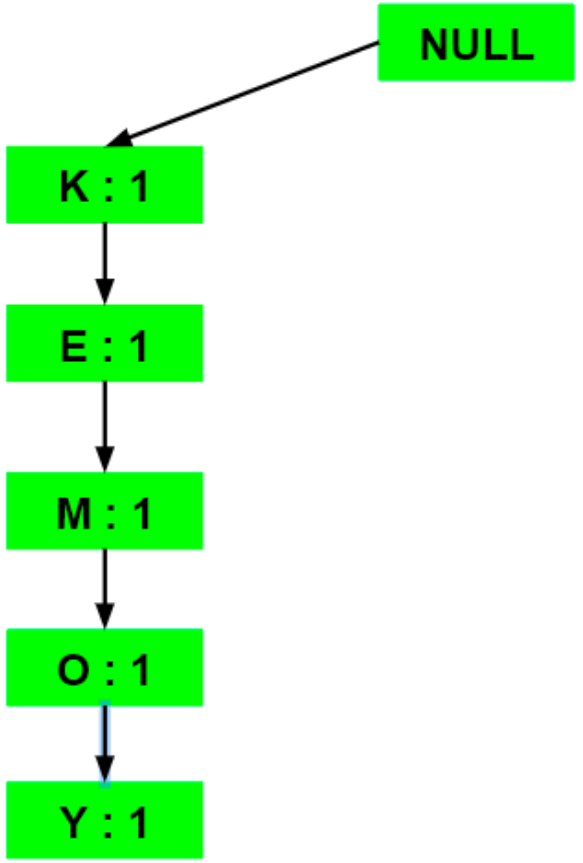
Frequent Patterns are

L = {K : 5, E : 4, M : 3, O : 3, Y : 3}

Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}

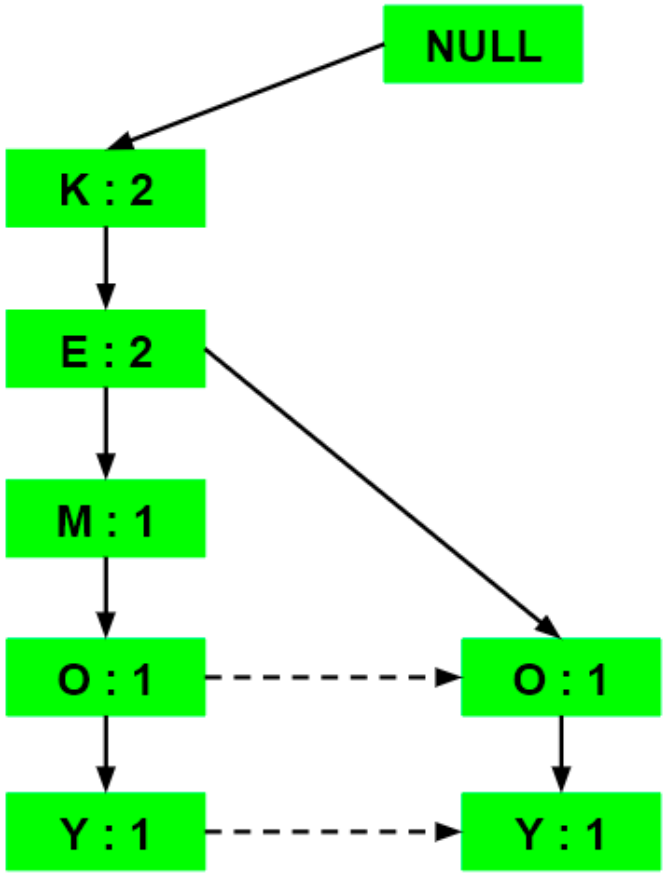
a) Inserting the set {K, E, M, O, Y}:

Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}



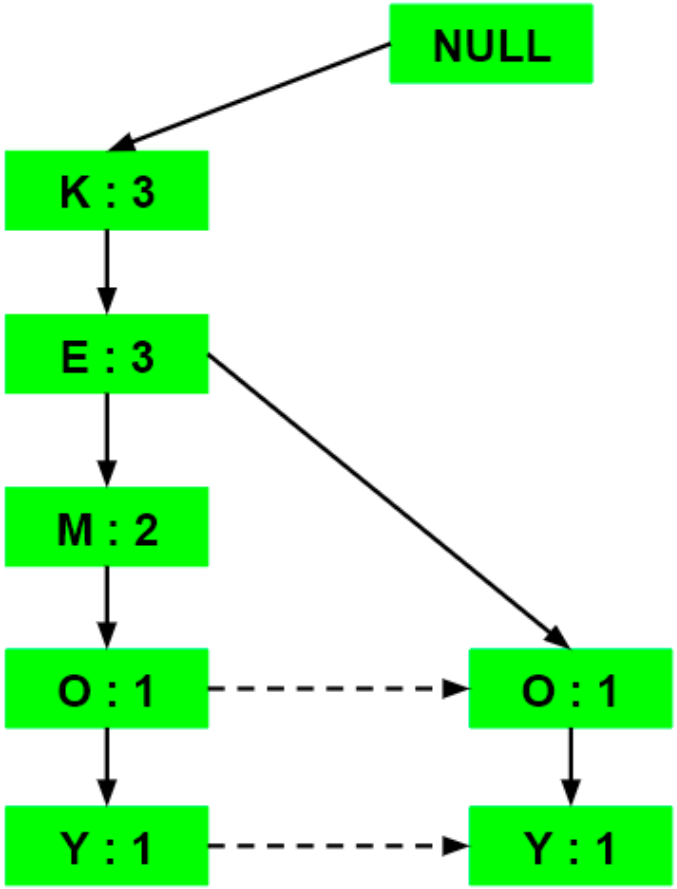
b) Inserting the set {K, E, O, Y}:

Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}



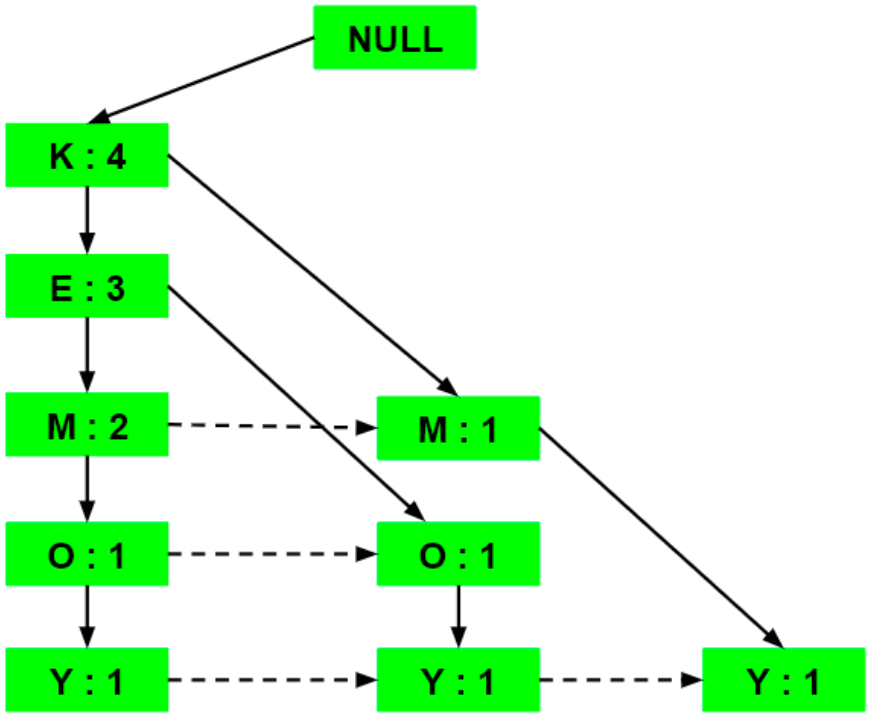
c) Inserting the set {K, E, M}:

Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}



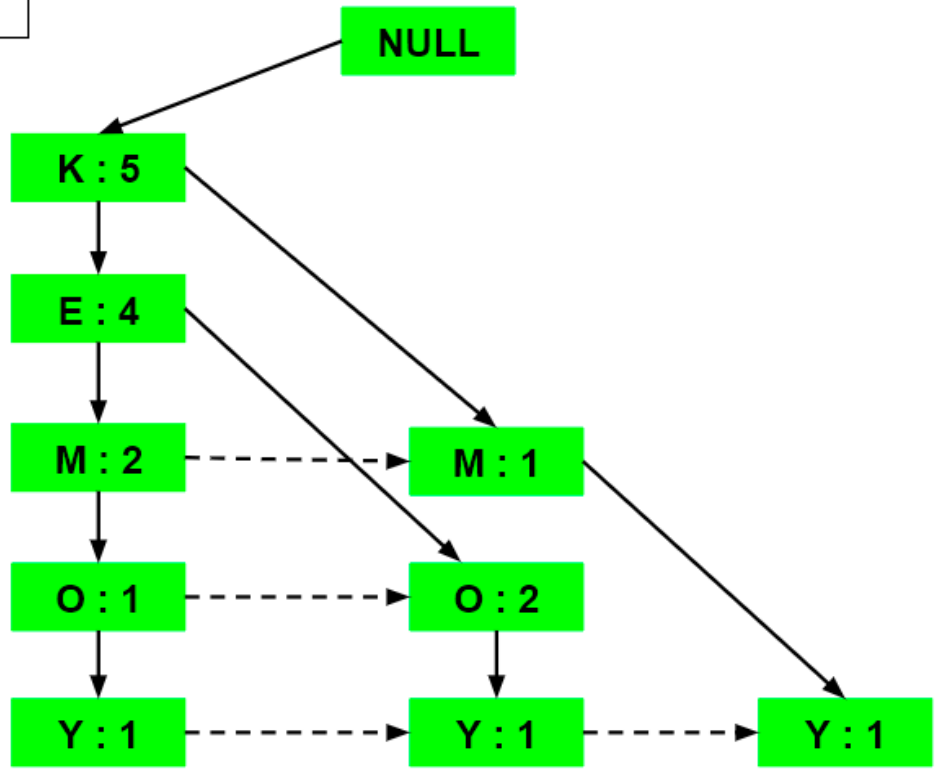
d) Inserting the set {K, M, Y}:

Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}



e) Inserting the set {K, E, O}:

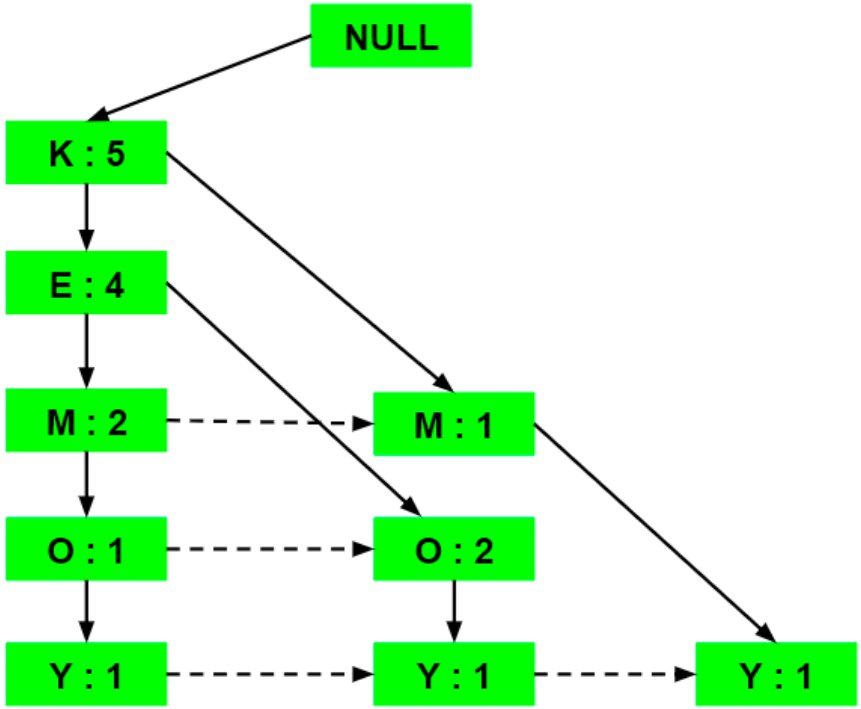
Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}



Compute Conditional Pattern Base

Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {K : 1}}
E	{K : 4}
K	

L = {K : 5, E : 4, M : 3, O : 3, Y : 3}



Built Conditional Frequent Pattern Tree

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

Generate Frequent Pattern rules

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

6.2.5 Mining Frequent Itemsets Using the Vertical Data Format

Table 6.3 The Vertical Data Format of the Transaction Data Set *D* of Table 6.1

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}

Table 6.4 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}

Table 6.5 3-Itemsets in Vertical Data Format

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

Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)

6.2.6 Mining Closed and Max Patterns

- **Item merging**:
  - If every transaction containing a frequent itemset X also contains an itemset Y but not any proper superset of Y , then X ∪ Y forms a frequent closed itemset and there is no need to search for any itemset containing X but no Y .
- **Sub-itemset pruning**:
  - If a frequent itemset 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 itemsets and thus can be pruned.

Mining Closed and Max Patterns

- **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.

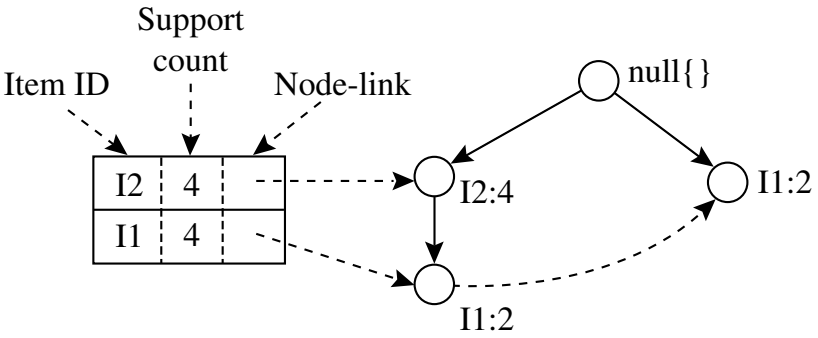
Another Example

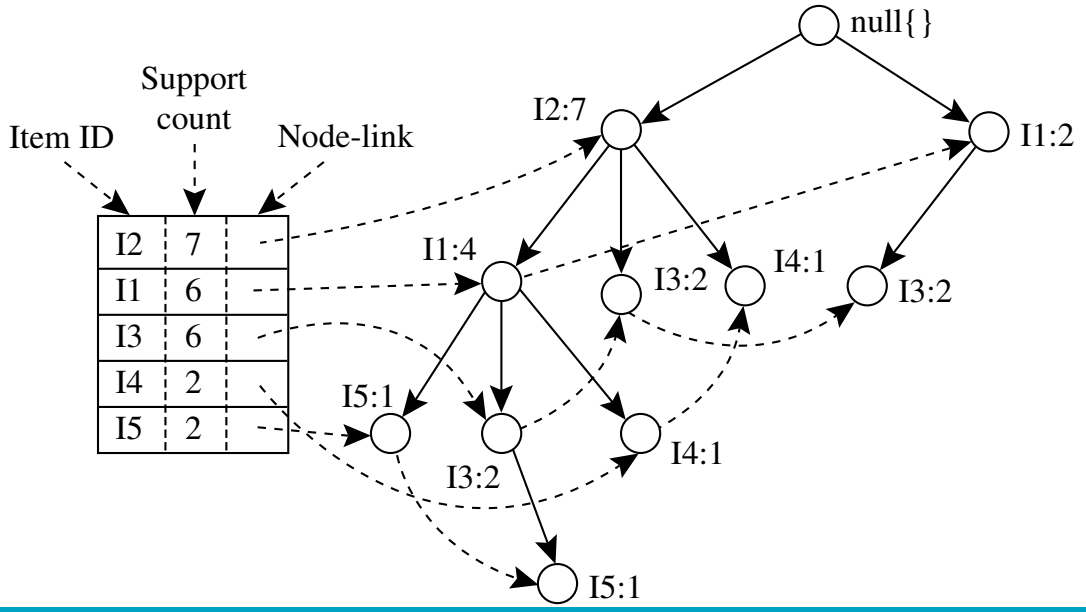
TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Another Example

Mining the FP-Tree by Creating Conditional (Sub-)Pattern Bases

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*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: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}





6.2.5 Mining Frequent Itemsets Using the Vertical Data Format

Table 6.3 The Vertical Data Format of the Transaction Data Set *D* of Table 6.1

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}

Table 6.4 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}

Table 6.5 3-Itemsets in Vertical Data Format

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

Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)

6.2.6 Mining Closed and Max Patterns

- **Item merging**:
  - If every transaction containing a frequent itemset X also contains an itemset Y but not any proper superset of Y , then X ∪ Y forms a frequent closed itemset and there is no need to search for any itemset containing X but no Y .
- **Sub-itemset pruning**:
  - If a frequent itemset 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 itemsets and thus can be pruned.

Mining Closed and Max Patterns

- **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.

Summary

- Basic concepts: association rules, support-confident framework, closed and max-patterns
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (Fpgrowth)
  - Vertical format approach

6.3 Which Patterns Are Interesting?—Pattern Evaluation Methods

- Most association rule mining algorithms employ a support–confidence framework.
- Although minimum support and confidence thresholds *help* weed out or exclude the exploration of a good number of uninteresting rules, many of the rules generated are still not interesting to the users.
- Unfortunately, this is especially true *when mining at low support thresholds or mining for long patterns*.
- This has been a major bottleneck for successful application of association rule mining.

6.3.1 Strong Rules Are Not Necessarily Interesting

- Whether or not a rule is interesting can be assessed either subjectively or objectively.
- Ultimately, if user judges subjectively – may differ from one person to another
- However, objectiveness can be used as one step toward the goal of weeding out uninteresting rules that would otherwise be presented to the user.
- *"How can we tell which strong association rules are really interesting?"*

Strong Rules Are Not Necessarily Interesting

- **Example: 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.
  - Association rule: a minimum support of 30% and a minimum confidence of 60%.
  - Rule is a strong association rule and would therefore be reported, since its **support value of 4000/ 10,000 = 40% and confidence value of 4000/6000 = 66%** satisfy the minimum support and minimum confidence thresholds, respectively.

Strong Rules Are Not Necessarily Interesting

- The following association rule is discovered:

$$buys(X, \text{“}computer\ games\text{”}) \Rightarrow buys(X, \text{“}videos\text{”})$$

$$[support = 40\%, confidence = 66\%].$$

- However, Rule 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 above Rule.
- Example also illustrates that the confidence of a rule $A \Rightarrow B$ can be deceiving.
- It does not measure the *real strength* (or lack of strength) of the *correlation* and *implication* between A and B.
- Hence, alternatives to the support–confidence framework can be useful in mining interesting data relationships.

6.3.2 From Association Analysis to Correlation Analysis

- Support and confidence measures are insufficient at filtering out uninteresting association rules.
- A correlation measure can be used to augment the support–confidence framework for association rules.
- This leads to *correlation rules* of the form

  $A \Rightarrow B$ [*support*, *confidence*, *correlation*]

- Many correlation measures are available:
  - Lift Measure
  - $\chi 2$ Measure

Lift Measure

- **Lift** is a simple correlation measure:
  - The occurrence of itemset A is **independent** of the occurrence of itemset B if $P(A \cup B) = P(A)P(B)$
  - Otherwise, itemsets A and B are **dependent** and **correlated** as events.
- The **lift** between the occurrence of A and B can be measured by computing

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

- Lift(A,B) <1 , A is *negatively correlated with* B
- Lift(A,B) >1, A and B are *positively correlated*
- Lift(A,B) = 1A and B are *independent* and no correlation
- In other words, *lift* of the association (or correlation) rule $A \Rightarrow B$ can be written as

$$lift(A, B) = \frac{P(B|A)}{P(B)} = conf(A \Rightarrow B) / Sup(B)$$

Lift: Example

2 × 2 Contingency Table Summarizing the Transactions with Respect to Game and Video Purchases

	game	$\overline{game}$	Σ_{row}
video	4000	3500	7500
$\overline{video}$	2000	500	2500
Σ_{col}	6000	4000	10,000

- *From table,* $P(\{game\}) = 0.60$, $P(\{video\}) = 0.75$, and $P(\{game, video\}) = 0.40$

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

- *Lift (game, video) < 1,* there is a negative correlation between the occurrence of {*game*} and {*video*}.
- The numerator is the likelihood of a customer purchasing both, while the denominator is what the likelihood would have been if the two purchases were completely independent.
- Such a negative correlation cannot be identified by a support–confidence framework.

χ2 measure

- To compute the χ2 value, take the squared difference between the observed and expected value for a slot (*A* and *B* pair) in the contingency table, divided by the expected value.
- This amount is summed for all slots of the contingency table. $\chi^2 = \Sigma \frac{(observed - expected)^2}{expected}$

Contingency Table

	game	$\overline{game}$	Σ_{row}
video	4000	3500	7500
$\overline{video}$	2000	500	2500
Σ_{col}	6000	4000	10,000

Expected Values

	game	$\overline{game}$	Σ_{row}
video	4000 (4500)	3500 (3000)	7500
$\overline{video}$	2000 (1500)	500 (1000)	2500
Σ_{col}	6000	4000	10,000

$$\chi^2 = \Sigma \frac{(observed - expected)^2}{expected} = \frac{(4000-4500)^2}{4500} + \frac{(3500-3000)^2}{3000} + \frac{(2000-1500)^2}{1500} + \frac{(500-1000)^2}{1000} = 555.6.$$

- χ2 >1, and the observed value of the slot (*game*, *video*) = 4000, which is less than the expected value of 4500, *buying game* and *buying video* are *negatively correlated*.
- This is consistent with the conclusion derived from the analysis of the *lift* measure

Another Example

	play-basketball	not play-basketball	sum (row)
eat-cereal	400	350	750
not eat-cereal	200	50	250
sum(col.)	600	400	1000

Association

- Example: Suppose one school may have the following statistics on #of students who may play basketball and/or eat cereal:

2-way contingen

	play-basketball	not play-basketball	sum (row)
eat-cereal	400	350	750
not eat-cereal	200	50	250
sum(col.)	600	400	1000

- Association rule mining may generate the following:
  - *play-basketball* $\Rightarrow$ *eat-cereal* [40%, 66.7%] (higher s & c)
- But this strong association rule is misleading: The overall % of students eating cereal is 75% > 66.7%, a more telling rule:
  - $\neg$ *play-basketball* $\Rightarrow$ *eat-cereal* [35%, 87.5%] (high s & c)

Lift

- Measure of dependent/correlated events: **lift**

$$lift(B, C) = \frac{c(B \rightarrow C)}{s(C)} = \frac{s(B \cup C)}{s(B) \times s(C)}$$

Lift is more telling than s & c

	B	¬B	Σ_{row}
C	400	350	750
¬C	200	50	250
$\Sigma_{col.}$	600	400	1000

- Lift(B, C) may tell how B and C are correlated
  - Lift(B, C) = 1: B and C are independent
  - > 1: positively correlated
  - < 1: negatively correlated
- For our example,

$$lift(B, C) = \frac{400/1000}{600/1000 \times 750/1000} = 0.89$$

$$lift(B, \neg C) = \frac{200/1000}{600/1000 \times 250/1000} = 1.33$$

Chi-Square

- Another measure to test correlated events: χ^2

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- General rules
  - $\chi^2 = 0$: independent
  - $\chi^2 > 0$: correlated, either positive or negative, so it needs additional test

χ^2 tells also better than s & c

	B	¬B	Σ_{row}
C	400 (450)	350 (300)	750
¬C	200 (150)	50 (100)	250
Σ_{col}	600	400	1000

Expected value

Observed value

- Now, $\chi^2 = \frac{(400-450)^2}{400} + \frac{(350-300)^2}{350} + \frac{(200-150)^2}{200} + \frac{(50-100)^2}{50} = 55.89$
- χ^2 shows B and C are negatively correlated since the expected value is 450 but the observed is only 400

Which is best? *lift* and $\chi 2$

- Null transactions: Transactions that contain neither B nor C

	B	¬B	Σ_{row}
C	100	1000	1100
¬C	1000	100000	101000
$\Sigma_{col.}$	1100	101000	102100

Which is best? *lift* and $\chi 2$

- Null transactions: Transactions that contain neither B nor C

	B	¬B	Σ_{row}
C	100	1000	1100
¬C	1000	100000	101000
$\Sigma_{col.}$	1100	101000	102100

null transactions

Which is best? *lift* and $\chi 2$

- Null transactions: Transactions that contain neither B nor C
- Let's examine the dataset D
  - BC (100) is much rarer than B¬C (1000) and ¬BC (1000), but there are many ¬B¬C (100000)
  - Unlikely B & C will happen together!

	B	¬B	Σ_{row}
C	100	1000	1100
¬C	1000	100000	101000
$\Sigma_{col.}$	1100	101000	102100

null transactions

Which is best? *lift* and $\chi 2$

- Null transactions: Transactions that contain neither B nor C
- Let's examine the dataset D
  - BC (100) is much rarer than B¬C (1000) and ¬BC (1000), but there are many ¬B¬C (100000)
  - Unlikely B & C will happen together!
- But, Lift(B, C) = 8.44 >> 1 (Lift shows B and C are strongly positively correlated!)

	B	¬B	$\sum_{row}$
C	100	1000	1100
¬C	1000	100000	101000
$\sum_{col.}$	1100	101000	102100

null transactions

Which is best? *lift* and $\chi 2$

- Null transactions: Transactions that contain neither B nor C
- Let's examine the dataset D
  - BC (100) is much rarer than B¬C (1000) and ¬BC (1000), but there are many ¬B¬C (100000)
  - Unlikely B & C will happen together!
- But, Lift(B, C) = 8.44 >> 1 (Lift shows B and C are strongly positively correlated!)

	B	¬B	Σ_{row}
C	100	1000	1100
¬C	1000	100000	101000
$\Sigma_{col.}$	1100	101000	102100

null transactions

Contingency table with expected values added

	B	¬B	Σ_{row}
C	100 (11.85)	1000	1100
¬C	1000 (988.15)	100000	101000
$\Sigma_{col.}$	1100	101000	102100

Which is best? *lift* and $\chi 2$

- Null transactions: Transactions that contain neither B nor C
- Let's examine the dataset D
  - BC (100) is much rarer than B¬C (1000) and ¬BC (1000), but there are many ¬B¬C (100000)
  - Unlikely B & C will happen together!
- But, Lift(B, C) = 8.44 >> 1 (Lift shows B and C are strongly positively correlated!)
- χ^2 = 670: Observed(BC) >> expected value (11.85)

	B	¬B	$\sum_{row}$
C	100	1000	1100
¬C	1000	100000	101000
$\sum_{col.}$	1100	101000	102100

null transactions

Contingency table with expected values added

	B	¬B	$\sum_{row}$
C	100 (11.85)	1000	1100
¬C	1000 (988.15)	100000	101000
$\sum_{col.}$	1100	101000	102100

Which is best? *lift* and $\chi 2$

- Null transactions: Transactions that contain neither B nor C
- Let's examine the dataset D
  - BC (100) is much rarer than B¬C (1000) and ¬BC (1000), but there are many ¬B¬C (100000)
  - Unlikely B & C will happen together!
- But, Lift(B, C) = 8.44 >> 1 (Lift shows B and C are strongly positively correlated!)
- χ^2 = 670: Observed(BC) >> expected value (11.85)
- *Too many null transactions may "spoil the soup"!*

	B	¬B	Σ_{row}
C	100	1000	1100
¬C	1000	100000	101000
$\Sigma_{col.}$	1100	101000	102100

null transactions

Contingency table with expected values added

	B	¬B	Σ_{row}
C	100 (11.85)	1000	1100
¬C	1000 (988.15)	100000	101000
$\Sigma_{col.}$	1100	101000	102100

6.3.3 A Comparison of Pattern Evaluation Measures

- Instead of using the simple support–confidence framework other measures, such as *lift* and $\chi 2$ – discloses more intrinsic patterns
  - How effective are these measures?
  - Should we also consider other alternatives?
- Other measures:
  - *all confidence*
  - *max confidence*
  - *Kulczynski*
  - *cosine*

Other measures

- **All-confidence:**
- Given two itemsets, A and B, the **all confidence** measure of A and B is defined as

$$all\_conf(A,B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}$$

- where $max\{sup(A), sup(B)\}$ is the maximum support of the Itemsets A and B.
- Thus, *all conf*(A,B) is also the minimum confidence of the two association rules related to A and B, namely, "$A \Rightarrow B$" and "$B \Rightarrow A$."
- **Max_Confidence:**
- Given two itemsets, A and B, the **max confidence** measure of A and B is defined as

$$max\_conf(A, B) = max\{P(A \mid B), P(B \mid A)\}$$

- The *max conf* measure is the maximum confidence of the two association rules, "$A \Rightarrow B$" and "$B \Rightarrow A$."

Other Measures

- **Kulczynski measure:**
- Given two itemsets, A and B, the **Kulczynski** measure of A and B (abbreviated as **Kulc**) is defined as

$$Kulc(A, B) = \frac{1}{2}(P(A|B) + P(B|A))$$

- The average of two conditional probabilities: the probability of itemset B given itemset A, and the probability of itemset A given itemset B.
- **Cosine measure:**
- Finally, given two itemsets, A and B, the **cosine** measure of A and B is defined as

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

- The *cosine* measure can be viewed as a *harmonized lift* measure
  - The two formulae are similar except that for cosine, the *square root* is taken on the product of the probabilities of A and B.

Comparison of Null-invariant Measures

- Not all null-invariant measures are created equal
- Which one is better?

2×2 Contingency Table for Two Items

	$milk$	$\overline{milk}$	Σ_{row}
$coffee$	mc	$\overline{m}c$	c
$\overline{coffee}$	$m\overline{c}$	$\overline{mc}$	$\overline{c}$
Σ_{col}	m	$\overline{m}$	Σ

Comparison of Null-invariant Measures

- Not all null-invariant measures are created equal
- Which one is better?

2-variable contingency table

	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m\neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	$AllConf$	Jaccard	$Cosine$	$Kulc$	$MaxConf$
D_1	10,000	1,000	1,000	100,000	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	0.01	0.01	0.10	0.5	0.99

Comparison of Null-invariant Measures

- Not all null-invariant measures are created equal
- Which one is better?

2-variable contingency table

	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m\neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

All 5 are null-invariant

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	$AllConf$	Jaccard	$Cosine$	$Kulc$	$MaxConf$
D_1	10,000	1,000	1,000	100,000	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	0.01	0.01	0.10	0.5	0.99

Subtle: They disagree on those cases

Comparison of Null-invariant Measures

- Not all null-invariant measures are created equal
- Which one is better?
  - D_4—D_6 differentiate the null-invariant measures

2-variable contingency table

	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m\neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

All 5 are null-invariant

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	$AllConf$	Jaccard	$Cosine$	$Kulc$	$MaxConf$
D_1	10,000	1,000	1,000	100,000	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	0.01	0.01	0.10	0.5	0.99

Subtle: They disagree on those cases

Comparison of Null-invariant Measures

- Not all null-invariant measures are created equal
- Which one is better?
  - D_4—D_6 differentiate the null-invariant measures
  - Kulc (Kulcyzynski 1927) holds firm and is in balance of both directional implications

2-variable contingency table

	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m\neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

All 5 are null-invariant

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	$AllConf$	Jaccard	$Cosine$	$Kulc$	$MaxConf$
D_1	10,000	1,000	1,000	100,000	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	0.01	0.01	0.10	0.5	0.99

Subtle: They disagree on those cases

Comparison of Null-invariant Measures

- IR (Imbalance Ratio): measure the imbalance of two itemsets A and B in rule implications:

$$IR(A,B) = \frac{|sup(A) - sup(B)|}{sup(A) + sup(B) - sup(A \cup B)}$$

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	Jaccard	*Cosine*	*Kulc*	IR
D_1	10,000	1,000	1,000	100,000	0.83	0.91	0.91	0
D_2	10,000	1,000	1,000	100	0.83	0.91	0.91	0
D_3	100	1,000	1,000	100,000	0.05	0.09	0.09	0
D_4	1,000	1,000	1,000	100,000	0.33	0.5	0.5	0
D_5	1,000	100	10,000	100,000	0.09	0.29	0.5	0.89
D_6	1,000	10	100,000	100,000	0.01	0.10	0.5	0.99

Comparison of Null-invariant Measures

- IR (Imbalance Ratio): measure the imbalance of two itemsets A and B in rule implications:

$$IR(A,B) = \frac{|s(A) - s(B)|}{s(A) + s(B) - s(A \cup B)}$$

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	Jaccard	*Cosine*	*Kulc*	IR
D_1	10,000	1,000	1,000	100,000	0.83	0.91	0.91	0
D_2	10,000	1,000	1,000	100	0.83	0.91	0.91	0
D_3	100	1,000	1,000	100,000	0.05	0.09	0.09	0
D_4	1,000	1,000	1,000	100,000	0.33	0.5	0.5	0
D_5	1,000	100	10,000	100,000	0.09	0.29	0.5	0.89
D_6	1,000	10	100,000	100,000	0.01	0.10	0.5	0.99

- Kulczynski and Imbalance Ratio (IR) together present a clear picture for all the three datasets D_4 through D_6
  - D_4 is neutral & balanced; D_5 is neutral but imbalanced
  - D_6 is neutral but very imbalanced

Example

	Buy Apple	Not Buy Apple	Row Total
Buy Orange	2	5	7
Not Buy Orange	6	7	13
Column Total	8	12	20

Lift & Chi-Square

Scenario	Apple/ Orange	Not Apple/ Orange	Apple/ Not Orange	Not Apple/ Not Orange	Lift	Chi-Square
1	2	5	6	7	0.714	0.586
2	2	5	6	13	0.929	0.022

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

$$\chi^2 = \Sigma \frac{(observed - expected)^2}{expected}$$

Null Invariant Measures

Measure	Definition	Range	Null-Invariant
$\chi^2(A,B)$	$\sum_{i,j=0,1} \frac{(s(a_i b_j) - s(a_i b_j))^2}{s(a_i b_j)}$	$[0, \infty]$	No
$Lift(A,B)$	$\frac{s(A \cup B)}{s(A) \times s(B)}$	$[0, \infty]$	No
$AllConf(A,B)$	$\frac{s(A \cup B)}{max\{s(A), s(B)\}}$	$[0,1]$	Yes
$Jaccard(A,B)$	$\frac{s(A \cup B)}{s(A)+s(B)-s(A \cup B)}$	$[0,1]$	Yes
$Cosine(A,B)$	$\frac{s(A \cup B)}{\sqrt{s(A) \times s(B)}}$	$[0,1]$	Yes
$Kulczynski(A,B)$	$\frac{1}{2}(\frac{s(A \cup B)}{s(A)} + \frac{s(A \cup B)}{s(B)})$	$[0,1]$	Yes
$MaxConf(A,B)$	$max\{\frac{s(A)}{s(A \cup B)}, \frac{s(B)}{s(A \cup B)}\}$	$[0,1]$	Yes

χ^2 and lift are not null-invariant

Jaccard, Cosine, AllConf, MaxConf, and Kulczynski are null-invariant measures

	Buy Apple	Not Buy Apple	Row Total
Buy Orange	2	5	7
Not Buy Orange	6	7	13
Column Total	8	12	20

					Interestingness Measure:						
Scenario	Apple/ Orange	Not Apple/ Orange	Apple/ Not Orange	Not Apple/ Not Orange	Lift	Chi-Square	All Confidence	Kulczynski	Max Confidence	Cosine	Jaccard
1	2	5	6	7	0.714	0.586	0.250	0.368	0.286	0.267	0.154
2	2	5	6	13	0.929	0.022	0.250	0.368	0.286	0.267	0.154

Unit - II

Mining Frequent Patterns, Associations, Correlations

(Contents: Text book 2 - Chapter 7)

Dr. R. Elakkiya
AP-III, SOC
SASTRA Deemed University

Advanced Frequent Pattern Mining

- Pattern Mining in Multi-Level, Multi-Dimensional Space
  - Mining Multi-Level Association
  - Mining Multi-Dimensional Association
  - Mining Quantitative Association Rules
  - Mining Rare Patterns and Negative Patterns
- Summary

Pattern Mining in Multi-Level, Multi-Dimensional Space

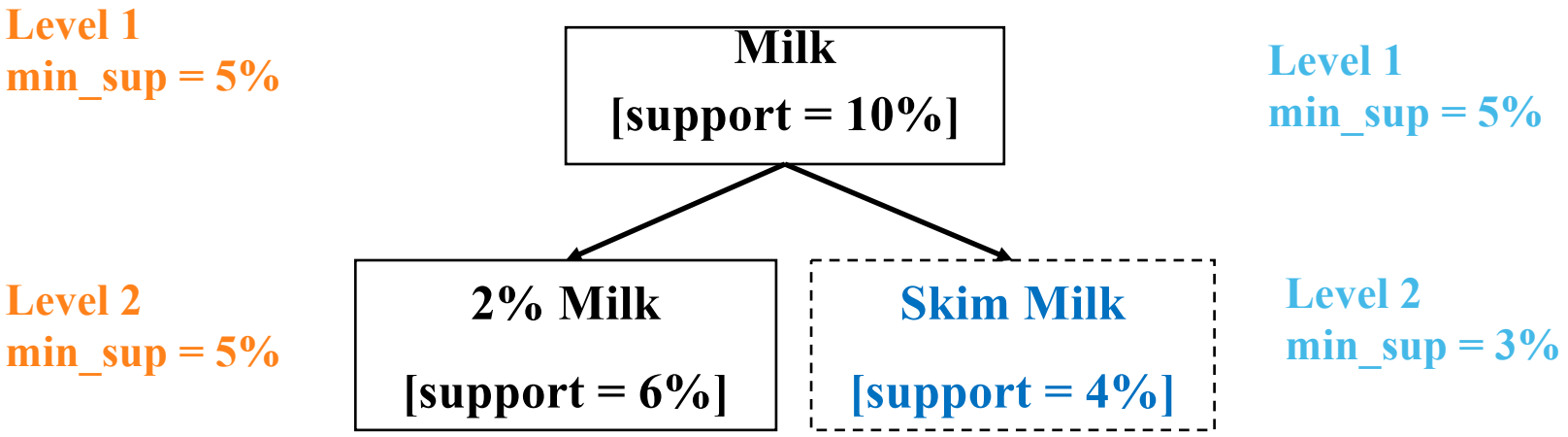
- focuses on methods for mining in multilevel, multidimensional space.
  - multilevel associations - concepts at different abstraction levels.
  - multi- dimensional associations - involve more than one dimension or predicate (e.g., rules that relate what a customer *buys* to his or her *age*).
  - quantitative association rules - numeric attributes that have an implicit ordering among values (e.g., *age*).
  - rare patterns - suggest interesting although rare item combinations.
  - negative patterns - show negative correlations between items.

Mining Multiple-Level Association Rules

- effective methods for mining patterns at multiple abstraction levels
- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support
- Exploration of *shared* multi-level mining

uniform support

reduced support



Multi-level Association: Flexible Support and Redundancy filtering

- Flexible min-support thresholds: Some items are more valuable but less frequent
  - Use non-uniform, group-based min-support
  - E.g., {diamond, watch, camera}: 0.05%; {bread, milk}: 5%; …
- Redundancy Filtering: Some rules may be redundant due to "ancestor" relationships between items
  - milk ⇒ wheat bread [support = 8%, confidence = 70%]
  - 2% milk ⇒ wheat bread [support = 2%, confidence = 72%]

  The first rule is an ancestor of the second rule
- A rule is *redundant* if its support is close to the "expected" value, based on the rule's ancestor

Mining Multi-Dimensional Association

- Single-dimensional rules:

  $buys(X, \text{“milk”}) \Rightarrow buys(X, \text{“bread”})$

- Multi-dimensional rules: ≥ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    $age(X,\text{”19-25”}) \wedge occupation(X,\text{“student”}) \Rightarrow buys(X, \text{“coke”})$
  - hybrid-dimension assoc. rules (*repeated predicates*)

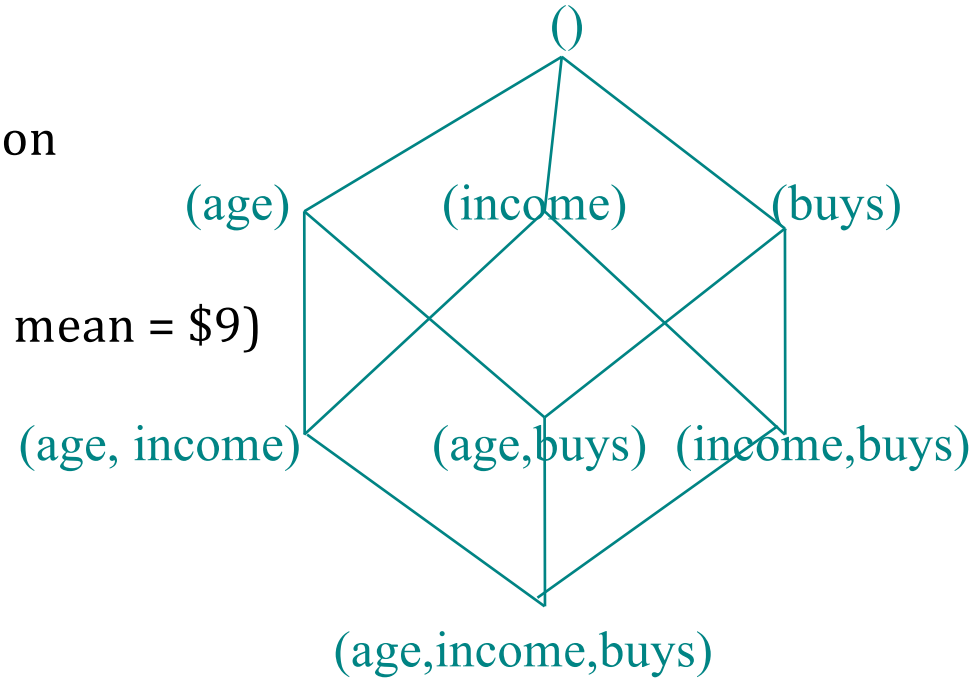
    $age(X,\text{”19-25”}) \wedge buys(X, \text{“popcorn”}) \Rightarrow buys(X, \text{“coke”})$

- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach
- Quantitative Attributes: Numeric, implicit ordering among values—discretization, clustering, and gradient approaches

Mining Quantitative Associations

Techniques can be categorized by how numerical attributes, such as age or salary are treated

1. Static discretization based on predefined concept hierarchies (data cube methods)
2. Dynamic discretization based on data distribution
3. Clustering: Distance-based association
   - One dimensional clustering then association
4. Deviation:
5. Sex = female => Wage: mean=$7/hr (overall mean = $9)



Quantitative Association Rules Based on Statistical Inference Theory

- Finding extraordinary and therefore interesting phenomena, e.g.,

  (Sex = female) => Wage: mean=$7/hr (overall mean = $9)

  - LHS: a subset of the population
  - RHS: an extraordinary behavior of this subset
- The rule is accepted only if a statistical test (e.g., Z-test) confirms the inference with high confidence
- Subrule: highlights the extraordinary behavior of a subset of the pop. of the super rule
  - E.g., (Sex = female) ^ (South = yes) => mean wage = $6.3/hr
- Two forms of rules
  - Categorical => quantitative rules, or Quantitative => quantitative rules
  - E.g., Education in [14-18] (yrs) => mean wage = $11.64/hr
- Open problem: Efficient methods for LHS containing two or more quantitative attributes

Negative and Rare Patterns

- Rare patterns: Very low support but interesting
  - E.g., buying Rolex watches
  - Mining: Setting individual-based or special group-based support threshold for valuable items
- Negative patterns
  - Since it is unlikely that one buys Ford Expedition (an SUV car) and Toyota Prius (a hybrid car) together, Ford Expedition and Toyota Prius are likely negatively correlated patterns
- Negatively correlated patterns that are infrequent tend to be more interesting than those that are frequent

Defining Negative Correlated Patterns (I)

- Definition 1 (support-based)
  - If itemsets X and Y are both frequent but rarely occur together, i.e.,

    $$sup(X \cup Y) < sup(X) * sup(Y)$$

  - Then X and Y are negatively correlated
- Problem: A store sold two needle 100 packages A and B, only one transaction containing both A and B.
  - When there are in total 200 transactions, we have

    $$s(A \cup B) = 0.005,\ s(A) * s(B) = 0.25,\ s(A \cup B) < s(A) * s(B)$$

  - When there are 10^5 transactions, we have

    $$s(A \cup B) = 1/10^5,\ s(A) * s(B) = 1/10^3 * 1/10^3,\ s(A \cup B) > s(A) * s(B)$$

  - Where is the problem? —Null transactions, i.e., the support-based definition is not null-invariant!

Defining Negative Correlated Patterns (II)

- Definition 2 (negative itemset-based)
  - X is a *negative itemset* if (1) X = $\bar{A}$ U B, where B is a set of positive items, and $\bar{A}$ is a set of negative items, $|\bar{A}| \geq 1$, and (2) $s(X) \geq \mu$
  - Itemsets X is negatively correlated, if

$$s(X) < \prod_{i=1}^{k} s(x_i), where\ x_i \in X,\ and\ s(x_i)\ is\ the\ support\ of\ x_i$$

- This definition suffers a similar null-invariant problem
- Definition 3 (Kulzynski measure-based) If itemsets X and Y are frequent, but $(P(X|Y) + P(Y|X))/2 < \epsilon$, where ϵ is a negative pattern threshold, then X and Y are negatively correlated.
- Ex. For the same needle package problem, when no matter there are 200 or 10^5 transactions, if $\epsilon = 0.01$, we have

  $(P(A|B) + P(B|A))/2 = (0.01 + 0.01)/2 < \epsilon$

SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 OF THE UGC ACT, 1956)
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

Dr. R. Elakkiya, AP-SoC, SASTRA Deemed University