UNIT III
ASSOCIATION RULE MINING

CONTENTS

3.1. BASIC CONCEPTS

Association rule mining is the data mining process of finding the rules that may govern associations and causal objects between sets of items.
So in a given transaction with multiple items, it tries to find the rules that govern how or why such items are often bought together. For example, peanut butter and jelly are often bought together because a lot of people like to make PB&J sandwiches.

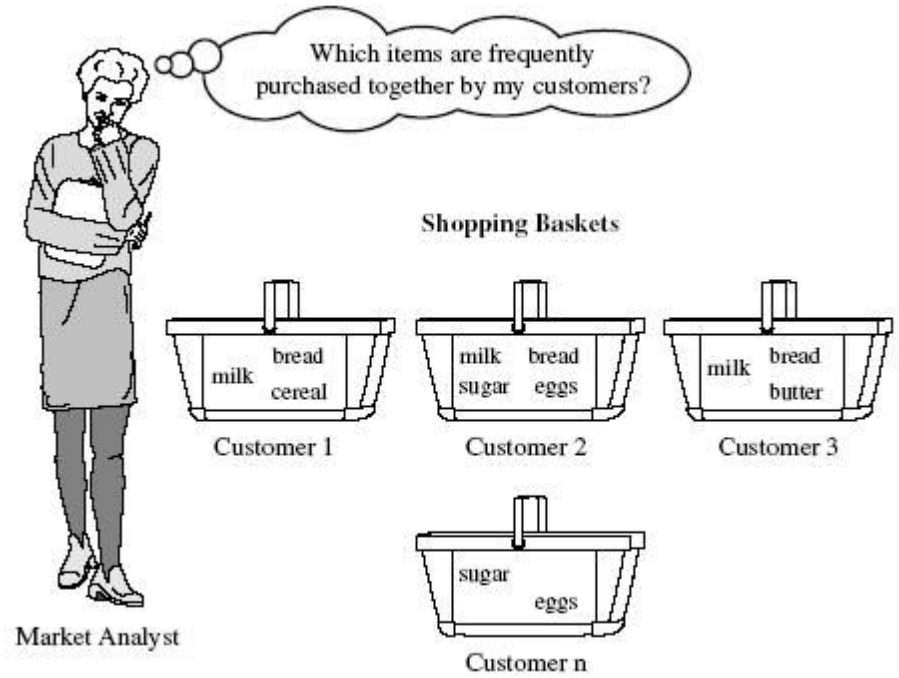
The main applications of association rule mining:

- Basket data analysis - is to analyze the association of purchased items in a single basket or single purchase as per the examples given above.
- Cross marketing - is to work with other businesses that complement your own, not competitors. For example, vehicle dealerships and manufacturers have cross marketing campaigns with oil and gas companies for obvious reasons.
- Catalog design - the selection of items in a business' catalog are often designed to complement each other so that buying one item will lead to buying of another. So these items are often complements or very related

3.2. FREQUENT ITEMSET MINING METHODS

Frequent pattern mining searches for recurring relationships in a given data set. It introduces the basic concepts of frequent pattern mining for the discovery of interesting associations and correlations between itemsets in transactional and relational databases.

Figure 3.1. Market Basket Analysis



An example of frequent itemset mining is Figure 3.1. market basket analysis. This process analyzes customer buying habits by finding associations between the different items that customers place in their "shopping baskets". The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket? Such information can lead to increased sales by helping retailers do selective marketing and plan their shelf space.

If we think of the universe as the set of items available at the store, then 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. For example, the information that

customers who purchase computers also tend to buy antivirus software at the same time is represented in Association Rule below:

Computer =>antivirus software [*support* = 2%; *confidence* =60%]

Rule support and

confidence are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules. A support of 2% for Association Rule (5.1) means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together. A confidence of 60% means that 60% of the customers who purchased a computer also bought the software. Association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold. Such thresholds can be set by users or domain experts. Additional analysis can be performed to uncover interesting statistical correlations between associated items.

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**.

 The set {*computer, antivirus software}* is a **2-itemset**.

 The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is also known, simply, as the **frequency, support count,** or **count** of the itemset.

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

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

 Rules that satisfy both a minimum support threshold (*min sup*) and a minimum confidence threshold (*min conf*) are called **Strong Association Rules.**

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

1. Find all frequent itemsets: By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min_sup*.

2. Generate strong association rules from the frequent itemsets: By definition, these rules must satisfy minimum support and minimum confidence.

3.3. The Apriori Algorithm:

Finding Frequent Itemsets Using Candidate Generation

The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset properties, as we shall see following. Apriori employs an iterative approach known as a *level-wise* search, where *k*-itemsets are used to explore (*k*+1)-itemsets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item,and collecting those items that satisfy minimum support. The resulting set is denoted *L*1.Next, *L*1

is used to find *L*2, the set of frequent 2-itemsets, which is used to find *L*3, and so on, until no more frequent *k*-itemsets can be found. The finding of each *Lk* 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, presented below, is used to reduce the search space. We will first describe this property, and then show an example illustrating its use. Apriori property: All nonempty subsets of a frequent itemset must also be frequent.

A two-step process is followed, consisting of join and prune actions

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

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

Table 4.1 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

Generating Association Rules from Frequent Itemsets

Once the frequent itemsets from transactions in a database *D* have been found, it is straightforward to generate strong association rules from them (where *strong* association rules satisfy both minimum support and minimum confidence). This can be done using Equation (5.4) for confidence, which we show again here for completeness:

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

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.coun ≥ min_sup}
(10) }
(11) return L = ∪k Lk;
```

3.4. FP-Growth Method: Mining Frequent Itemsets without Candidate Generation

FP-growth adopts a *divide-and-conquer* strategy as follows. First, it compresses the database representing frequent items into a frequent-pattern tree, or FP-tree, which retains the itemset association information. It then divides the compressed database into a set of *conditional databases* (a special kind of projected database), each associated with one frequent item or "pattern fragment," and mines each such database separately.

Example FP-growth (finding frequent itemsets without candidate generation). We re-examine the mining of transaction database, *D*, Figure 4.2 using the frequent pattern growth approach.

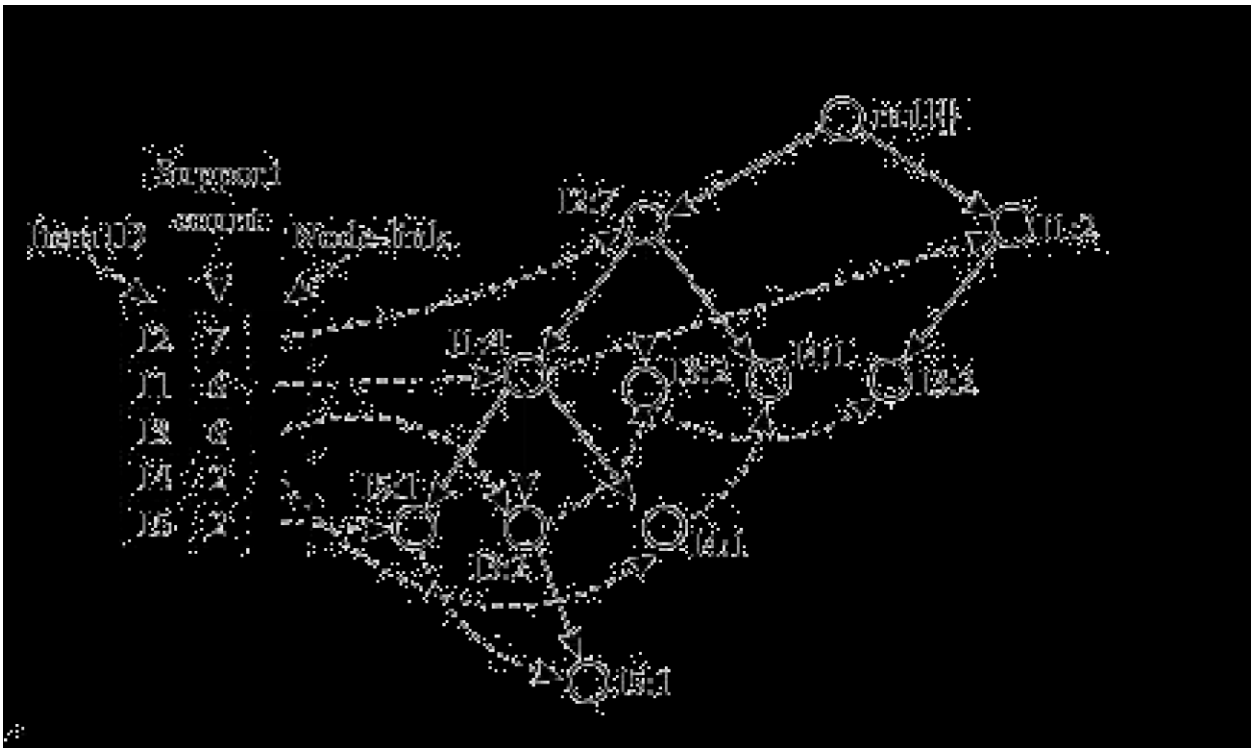


Figure 3.2 FP growth method

3.5. Mining Various Kinds of Association Rules

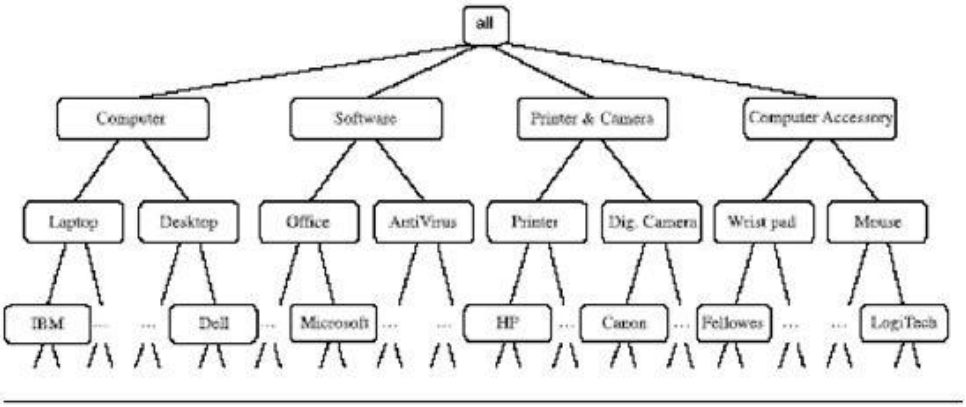
Mining Multilevel Association Rules

For many applications, it is difficult to find strong associations among data items at low or primitive levels of abstraction due to the sparsity of data at those levels. Strong associations discovered at high levels of abstraction may represent commonsense knowledge. Moreover, what may represent common sense to one user may seem novel to another. Therefore, data mining systems should provide capabilities for mining association rules at multiple levels of abstraction,

with sufficient flexibility for easy traversal among different abstraction spaces. Let's examine the following example. Mining multilevel association rules. Suppose we are given the task-relevant set of transactional data in Table for sales in an *AllElectronics* store, showing the items purchased for each transaction. The concept hierarchy for the items is shown in Figure 4.3. A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher level, more general concepts. Data can be generalized by replacing low-level conceptswithin the data by their higher-level concepts, or *ancestors*, from a concept hierarchy.

Figure 3.3. AllElectronics store

TID	Items Purchased
T100	IBM-ThinkPad-T40/2373, HP-Photosmart-7660
T200	Microsoft-Office-Professional-2003, Microsoft-Plus!-Digital-Media
T300	Logitech-MX700-Cordless-Mouse, Fellowes-Wrist-Rest
T400	Dell-Dimension-XPS, Canon-PowerShot-S400
T500	IBM-ThinkPad-R40/P4M, Symantec-Norton-Antivirus-2003
...	...



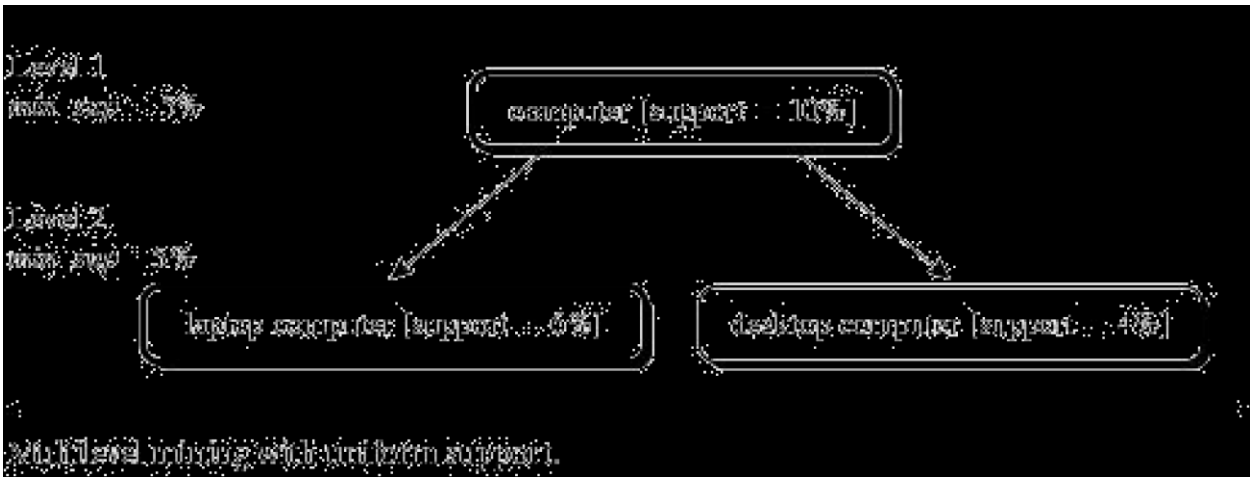
Association rules generated from mining data at multiple levels of abstraction are called multiple-level or multilevel association rules. Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework. In general, a topdown strategy is employed, where counts are accumulated for the calculation of frequent itemsets at each concept level, starting at the concept level 1 and working downward in the hierarchy toward the more specific concept levels, until no more frequent itemsets can be found. For each level, any algorithm for discovering frequent itemsets may be used, such as Apriori or

its variations.

Using uniform minimum support for all levels (referred to as uniform support):

The same minimum support threshold is used when mining at each level of abstraction. For example, a minimum support threshold of 5% is used throughout (e.g., for mining from *"computer"* down to *"laptop computer"*). Both *"computer"* and *"laptop computer"* are found to be frequent, while *"desktop computer"* is not. When a uniform minimum support threshold is used, the search procedure is simplified. The method is also simple in that users are required to specify only one minimum support threshold. An Apriori-like optimization technique can be adopted, based on the knowledge that an ancestor is a superset of its descendants: The search avoids examining itemsets containing any item whose ancestors do not have minimum support.

Figure 3.4.



Using reduced minimum support at lower levels (referred to as reduced support): Each level of abstraction has its own minimum support threshold. The deeper the level of abstraction, the smaller the corresponding threshold is. For example, in Figure, the minimum support thresholds for levels 1 and 2 are 5% and 3%, respectively. In this way, *"computer," "laptop computer,"* and *"desktop computer"* are all considered frequent.

Using item or group-based minimum support (referred to as group-based support):

Because users or experts often have insight as to which groups are more important than others, it is sometimes more desirable to set up user-specific, item, or group based minimal support thresholds when mining multilevel rules. For example, a user could set up the minimum support thresholds based on product price, or on items of interest, such as by setting particularly low

support thresholds for *laptop computers* and *flash drives* in order to pay particular attention to the association patterns containing items in these categories.

Mining Multidimensional Association Rules from Relational Databases and DataWarehouses

We have studied association rules that imply a single predicate, that is, the predicate *buys*. For instance, in mining our *AllElectronics* database, we may discover the Boolean association rule

$$buys(X, \text{“}digital\ camera\text{”}) \Rightarrow buys(X, \text{“}HP\ printer\text{”}).$$

Following the terminology used in multidimensional databases, we refer to each distinct predicate in a rule as a dimension. Hence, we can refer to Rule above as a single dimensional or intra dimensional association rule because it contains a single distinct predicate (e.g., *buys*)with multiple occurrences (i.e., the predicate occurs more than once within the rule). As we have seen in the previous sections of this chapter, such rules are commonly mined from transactional data.

Considering each database attribute or warehouse dimension as a predicate, we can therefore mine association rules containing *multiple* predicates, such as

$$age(X, \text{“}20...29\text{”}) \wedge occupation(X, \text{“}student\text{”}) \Rightarrow buys(X, \text{“}laptop\text{”}).$$

Association rules that involve two or more dimensions or predicates can be referred to as multidimensional association rules. Rule above contains three predicates (*age, occupation*, and *buys*), each of which occurs *only once* in the rule. Hence, we say that it has no repeated predicates. Multidimensional association rules with no repeated predicates are called inter dimensional association rules. We can also mine multidimensional association rules with repeated predicates, which contain multiple occurrences of some predicates. These rules are called hybrid-dimensional association rules. An example of such a rule is the following, where the predicate *buys* is repeated:

$$age(X, \text{“}20...29\text{”}) \wedge buys(X, \text{“}laptop\text{”}) \Rightarrow buys(X, \text{“}HP\ printer\text{”})$$

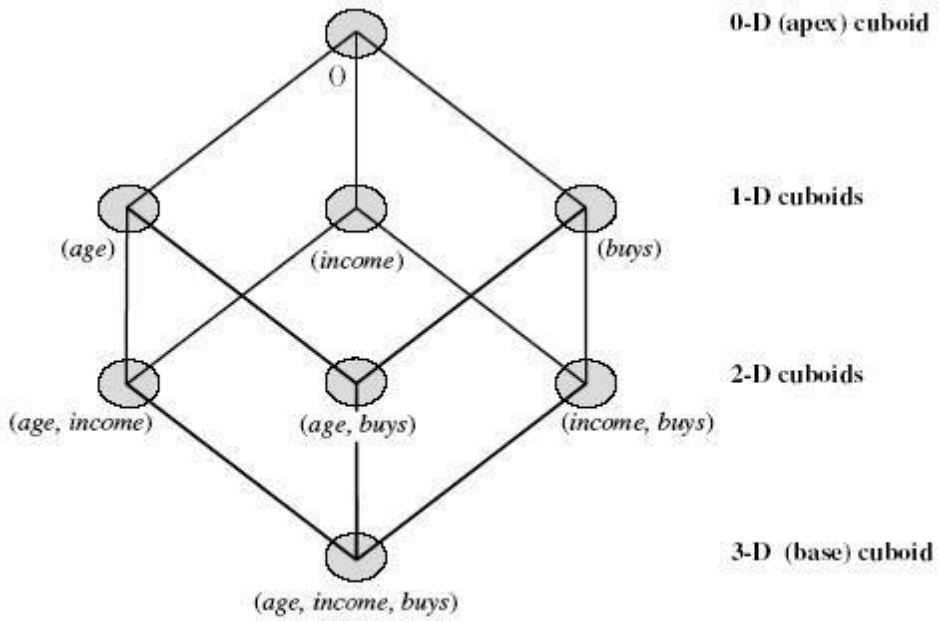
Note that database attributes can be categorical or quantitative. Categorical attributes have a finite number of possible values, with no ordering among the values (e.g., *occupation, brand, color*). Categorical attributes are also called nominal attributes, because their values are “names of things.” Quantitative attributes are numeric and have an implicit ordering among values (e.g.,

age, *income*, *price*). Techniques for mining multidimensional association rules can be categorized into two basic approaches regarding the treatment of quantitative attributes.

Mining Multidimensional Association Rules Using Static Discretization of Quantitative Attributes

Quantitative attributes, in this case, are discretized before mining using predefined concept hierarchies or data discretization techniques, where numeric values are replaced by interval labels. Categorical attributes may also be generalized to higher conceptual levels if desired. If the resulting task-relevant data are stored in a relational table, then any of the frequent itemset mining algorithms we have discussed can be modified easily so as to find all frequent predicate sets rather than frequent itemsets. In particular, instead of searching on only one attribute like *buys*, we need to search through all of the relevant attributes, treating each attributevalue pair as an itemset.

Figure 3.5.



Lattice of cuboids, making up a 3-D data cube. Each cuboid represents a different group-by. The base cuboid contains the three predicates *age*, *income*, and *buys*.

Mining Quantitative Association Rules

Quantitative association rules are multidimensional association rules in which the numeric

attributes are *dynamically* discretized during the mining process so as to satisfy some mining criteria, such as maximizing the confidence or compactness of the rules mined. In this section, we focus specifically on how to mine quantitative association rules having two quantitative attributes on the left-hand side of the rule and one categorical attribute on the right-hand side of the rule. That is,

$$A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$$

where *Aquan*1 and *Aquan*2 are tests on quantitative attribute intervals (where the intervals are dynamically determined), and *Acat* tests a categorical attribute from the task-relevant data. Such rules have been referred to as two-dimensional quantitative association rules, because they contain two quantitative dimensions. For instance, suppose you are curious about the association relationship between pairs of quantitative attributes, like customer age and income, and the type of television (such as *high-definition TV,* i.e., *HDTV*) that customers like to buy. An example of such a 2-D quantitative association rule is

Binning: Quantitative attributes can have a very wide range of values defining their domain. Just think about how big a 2-D grid would be if we plotted *age* and *income* as axes, where each possible value of *age* was assigned a unique position on one axis, and similarly, each possible value of *income* was assigned a unique position on the other axis! To keep grids down to a manageable size, we instead partition the ranges of quantitative attributes into intervals. These intervals are dynamic in that they may later be further combined during the mining process. The partitioning process is referred to as binning, that is, where the intervals are considered "bins."

Three common binning strategies area as follows:

Equal-width binning - the interval size of each bin is the same

Equal frequency binning - each bin has approximately the same number of tuples assigned to it

Clustering-based binning – clustering is performed on the quantitative attribute to group neighboring points

Finding frequent predicate sets: Once the 2-D array containing the count distribution for each category is set up, it can be scanned to find the frequent predicate sets (those satisfying minimum support) that also satisfy minimum confidence. Strong association rules can then be generated from these predicate sets, using a rule generation algorithm.

Clustering the association rules: The strong association rules obtained in the previous step are then mapped to a 2-D grid. Figure 5.14 shows a 2-D grid for 2-D quantitative association rules predicting the condition *buys*(*X*, "*HDTV*") on the rule right-hand side, given the quantitative attributes *age* and *income*. The four Xs correspond to the rules

$$age(X, 34) \wedge income(X, \text{"31}K\text{...40}K\text{"}) \Rightarrow buys(X, \text{"}HDTV\text{"}) \quad (5.16)$$
$$age(X, 35) \wedge income(X, \text{"31}K\text{...40}K\text{"}) \Rightarrow buys(X, \text{"}HDTV\text{"}) \quad (5.17)$$
$$age(X, 34) \wedge income(X, \text{"41}K\text{...50}K\text{"}) \Rightarrow buys(X, \text{"}HDTV\text{"}) \quad (5.18)$$
$$age(X, 35) \wedge income(X, \text{"41}K\text{...50}K\text{"}) \Rightarrow buys(X, \text{"}HDTV\text{"}). \quad (5.19)$$

"*Can we find a simpler rule to replace the above four rules?*" Notice that these rules are quite "close" to one another, forming a rule cluster on the grid. Indeed, the four rules can be combined or "clustered" together to form the following simpler rule, which subsumes and replaces the above four rules:

3.6. CORRELATION ANALYSIS

Most association rule mining algorithms employ a support-confidence framework. Often, many interesting rules can be found using low support thresholds. Although minimum support and confidence thresholds *help* weed out or exclude the exploration of a good number of uninteresting rules, many rules so 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 one of the major bottlenecks for successful application of association rule mining.

Strong Rules Are Not Necessarily Interesting: An Example

Whether or not a rule is interesting can be assessed either subjectively or objectively. Ultimately, only the user can judge if a given rule is interesting, and this judgment, being subjective, may differ from one user to another. However, objective interestingness measures, based on the statistics "behind" the data, can be used as one step toward the goal of weeding out uninteresting rules from presentation to the user.

The support and confidence measures are insufficient at filtering out uninteresting association rules. To tackle this weakness, a correlation measure can be used to augment the support confidence framework for association rules. This leads to *correlation rules* of the form

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

That is, a correlation rule is measured not only by its support and confidence but also by the correlation between itemsets *A* and *B*. There are many different correlation measures from which to choose. In this section, we study various correlation measures to determine which would be good for mining large data sets.

3.7. Constraint Based Association Mining

A data mining process may uncover thousands of rules from a given set of data, most of which end up being unrelated or uninteresting to the users. Often, users have a good sense of which "direction" of mining may lead to interesting patterns and the "form" of the patterns or rules they would like to find. Thus, a good heuristic is to have the users specify such intuition or expectations as *constraints* to confine the search space. This strategy is known as constraintbased mining. The constraints can include the following:

Knowledge type constraints : these specify the type of knowledge to be mined, such as association or correlation

Data constraints: these specify the set of task-relevant data

Dimension/ level constraints : these specify the desired dimensinos (or attributes) of the data, or levels of the concept hierarchies, to be used in mining

Interestingness, constraints: those specify thresholds on statistical measures of rule interestingness, such as support, confidence, and correlation

Rule constraints: these specify the form of rules to be mined. Such constraints may be expressed as meta rules (rules templates)as the maximum or minimum number of predicates that can occur in the rule antecedent or consequent, or as relationships among attributes, attribute values and or aggregates

Metarule-Guided Mining of Association Rules

"How are metarules useful?" Metarules allow users to specify the syntactic form of rules that they are interested in mining. The rule forms can be used as constraints to help improve the efficiency of the mining process. Metarules may be based on the analyst's experience, expectations, or intuition regarding the data or may be automatically generated based on the database schema.

Metarule-guided mining:- Suppose that as a market analyst for *AllElectronics,* you have access to the data describing customers (such as customer age, address, and credit rating) as well as the list of customer transactions. You are interested in finding associations between customer traits and the items that customers buy. However, rather than finding *all* of the association rules reflecting these relationships, you are particularly interested only in determining which pairs of customer traits promote the sale of office software.A metarule can be used to specify this information describing the form of rules you are interested in finding. An example of such a metarule is

$$P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{“}office\ software\text{”}),$$

where *P*1 and *P*2 are predicate variables that are instantiated to attributes from the given database during the mining process, *X* is a variable representing a customer, and *Y* and *W* take on values of the attributes assigned to *P*1 and *P*2, respectively. Typically, a user will specify a list of attributes to be considered for instantiation with *P*1 and *P*2. Otherwise, a default set may be used.

Constraint Pushing: Mining Guided by Rule Constraints

Rule constraints specify expected set/subset relationships of the variables in the mined rules, constant initiation of variables, and aggregate functions. Users typically employ their knowledge of the application or data to specify rule constraints for the mining task. These rule constraints may be used together with, or as an alternative to, metarule-guided mining. In this section, we examine rule constraints as to how they can be used to make the mining process more efficient. Let’s study an example where rule constraints are used to mine hybrid-dimensional association rules.

3.8. QUESTION BANK

PART A

1. What are the applications of association rule mining?
2. What is association rule mining? (Apr 2013)
3. List out the classification of association rules based on the types of values.
4. List out the classification of association rules based on the data. (Nov 2010)
5. List out the classification of association rules based on level of abstractions.
6. List out the classification of association rules based on various extensions.
7. What is constraint based rule mining?
8. Describe FP-growth. (Nov 2011)
9. What are distance based association rules?
10. Define confidence.
11. How is association rules mined from large databases? (Apr 2012)
12. What is the classification of association rules based on various criteria?
13. Discuss the concepts of frequent itemset, support & confidence.
14. What are the advantages of dimensional modelling?
15. Define multilevel association rules. (Apr 2012)
16. Define dimensional modelling**.**
17. Define anti-monotone property.
18. Define Iceberg query.
19. How to generate association rules from frequent item sets?
20. Define constraint-based association mining.

PART B

1. Explain the Apriori algorithm for discovering frequent itemsets for mining boolean association rules. (Nov '09,Nov'10)
2. Explain the FP-growth algorithm for discovering frequent itemsets without candidate generation.
3. Write notes on mining multilevel association rules. (Nov 2011)
4. Describe briefly about mining multidimensional association rules from relational databases and data Warehouses. (Apr 2012)
5. Describe how to mine multidimensional association rules using static discretization of quantitative attributes.

3.9 .REFERENCES

en.wikipedia.org/wiki/Association_rule_learning