**UNIT III**

**ASSOCIATION RULES:** Basic algorithms, Parallel and distributed algorithms, Incremental rules, advanced association rule techniques, measuring the quality of rules.

**Objectives:**

Identify the necessity of correlation analysis for association mining.

**3.1 Basic algorithms**

**Mining Association Rules in Large Databases**

Association rule mining searches for interesting relationships among items in a given data set.

**Market Basket Analysis**: Market Basket Analysis used to plan marketing or advertising strategies as well as catalog Design. Market Basket Analysis may help manages to designers different store layouts. In one strategy, items that are frequently purchased together can be placed in close proximity in order to encourage sale of such items together. In second strategy, by keeping the frequently purchased items together at opposite ends of the store may entice customers whose purchase such items to pick up other along the way. Association rules are considered as minimum support threshold and minimum confidence threshold.

Support: Percentage of transactions containing A that also defined like P(AUB).

Confidence: Percentage of transactions containing A that also contains B, defined like P(B/A).

Rules that satisfy both minimum support and confidence can be considered as strong. A set of items is referred as item set, A item set that contains K items called K-item set. An item set which satisfies minimum support can be called as frequent item set.

Association rule mining is a two step process:

Find all frequent item sets: Item sets will occur frequently has to be found i.e., item sets that satisfy predetermined minimum support count.

Generate strong Association rules from the frequent item sets: Rules that satisfy minimum support and minimum confidence.

Association Rule Mining: Classification of Association rules based on the types of values handled in the rule. If a rule concerns associations between presence and absence of items called Boolean association rule.

Association rule defined among quantitative items called quantitative association rule.

Age(,”30…39”) ^ income(,”42K…48K”) => buys(X,”TV”)

Age and income are quantitative attributes.

Based on the dimensions of data involved in the rule: Items or attributes in an association rule refers only one dimension called single dimensional association rule.

Buys(,”computer”) => buys(,”financial management S/W”)

Based on levels of abstractions involved in the rule set: Some association rules can be defined at different levels of abstraction.

Based on various extensions to association mining: Association mining can be extended to correlation analysis, where the absence or presence of correlated items can be identified. Mining Single dimensional Boolean Association rules from Transactional Databases.

**Single Dimensional Association Rules (Apriori Algorithm)**

Apriori Algorithm uses prior knowledge of frequent item set properties. Apriori employs an iterative approach called level wise search where k-item sets are used to explore (k+1) item sets.

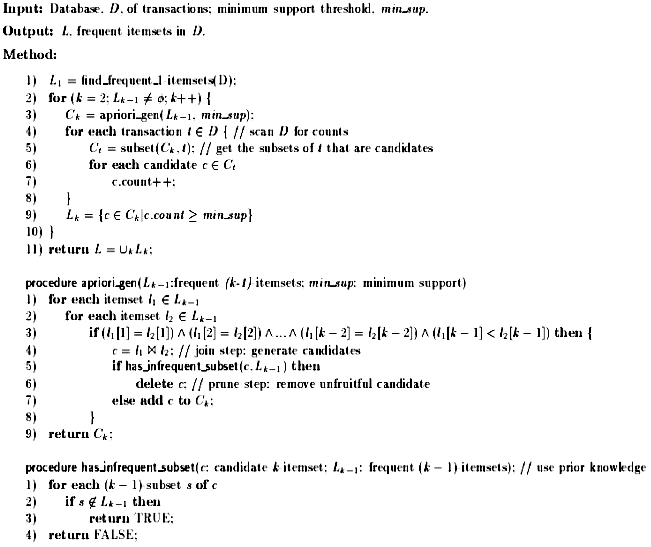
To reduce search space Apriori property is used All non empty sub sets of a frequent item set must be frequent. Apriori consists of two steps called join and prune.

The Prune Step: Ck is a super set of Lk, all its members may or may not be frequent. Non frequent items are removed with Apriori priority. If any (k-1) subset of a candidate k item set is not in Lk-1, then candidate cannot be frequent so it can be removed from Ck.

The join Step: To find LK, a set of candidate k-item sets is generated by joining Lk-1 with itself candidate set is denoted by Ck.

|  |  |
| --- | --- |
| T ID | List of Items |
| T1 | I1,I2,I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

Algorithm 6.2.1 (Apriori) Find frequent item sets using an iterative level-wise approach.





Generating association rules from frequent item sets

confidence(A => B) = Prob(B|A) =

where support(A U B) is the number of transactions containing the item sets

A U B, and support(A) is the number of transactions containing the itemset A.

- For each frequent item set, l, generate all non-empty subsets of l.

-For every non-empty subset s, of l, output the rule, s => (l – s)"

if support(l)/support(s) ≥ min conf, where min conf is the minimum confidence threshold.

I1˄ I2 =>I5 Confidence =2/4=50%

**Improvements of Efficiency of Apriori:**

Hash based Technique: It will reduce the size of the candidate k-item sets. All z-item sets hashed into the different buckets and then bucket counts are increased. An item set whose bucket count is below the support threshold can not be frequent and it should be removed from the candidate set.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

Create a hash table using hash function

H(x,y)=((order of x) \* 10 + (order of y) ) mod 7

Transaction Reduction: A transaction that does not contain any frequent k- item sets cannot contain any frequent (k+1) item sets. All such transactions need to be removed.

Partitioning: It needs two database scans to mine frequent item sets. In phase I, sub divides the transactions into non overlapping partitions. Based on minimum support count all frequent item sets with in a partition are found and those are referred as local frequent item sets. The collection of all local frequent item sets forms a global candidate item sets.

In phase II, actual support count of each candidate is assessed and global frequent items are determined.

Sampling: pick up a random sample S of items from D and then search for frequent item sets in S instead of D. To avoid missing of global frequent item sets support threshold has to be reduced then minimum support count.

Dynamic item set counting: Database is partitioned into blocks marked by start points. New candidate item sets added at any start point. It will reduce number of database scans.

Mining frequent item set without candidate generation: First scan the database to derive the set of frequent items and their support count. The set of frequent items is stored in descending order to create a root of the tree and label it with Null, items in each transaction are processed in descending in order and branch is created for each transaction. The count of each node along a common prefix is increased by 1 and nodes for the following items are created and linked accordingly.

Null {}

I1:2

I3 : 2

I2:7

I1:4

I3:2

I4:1

I4:1

I3:2

I5:1

I5:1

I5 is the last item in L, I5 occurs in two branches the paths formed by these branches are (I2 I1 I5 :1) and (I2 I1 I3 I5 : 1). The prefixes (I2 I1 :1) and (I2 I1 I3 :1) from a conditional pattern base. Condition FP-tree contains a single path (I2:2,I1:2):I3 is not include because its support count is less than minimum support count. Single path generates all the combinations of frequent patterns I2 I5:2, I1 I5:2, I2I1I5:2.

|  |  |  |  |
| --- | --- | --- | --- |
| item | Conditional pattern base | Conditional FP tree | Frequent pattern generated |
| I5 | {(I2 I1 :1)(I2 I1 I3:1)} | {(I2:2)(I1:2)} | I2I1I5:2, I2I5:2, I2I5:2 |
| I4 | {(I2 I1 :1)(I2:4)} | (I2:2) | I2I4:2 |
| I3 | {(I2 I1 :1)(I2:2)(I1:2)} | (I2:4, I1:2)(I1:2) | I2I3:4, I1I3:2, I2I1I3:2 |
| I1 | I2:4 | I2:4 | I2I1:4 |

**Iceberg Queries**: An iceberg query computes an aggregate function over an attribute or set of attributes in order to find aggregate values above some specific threshold.

Select R.a\_1, R.a\_2,…, R.a\_n,agg\_f(R.b)

From relation R

Group by R.a\_1,R.a\_2…R.a\_n

Having agg\_f(R.b) >=threshold.

Within the large quantity of input data tuples, relatively small number of tuples will satisfy the having clause. The output is seen as tip of iceberg and iceberg is the set of input data.

**Approaches to mining multilevel association rules:**

|  |  |
| --- | --- |
| TID | Items Purchased |
| T1  T2  T3  T4  T5 | IBM home computer, Sony b/w printer  Microsoft educational software, Microsoft financial management software  Logitech mouse computer-accessory, Ergo-way wrist pad computer-accessory  IBM home computer, Microsoft financial management software  IBM home computer |

A Concept hierarchy for All Electronics Computer items

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. In this case, the search procedure is simplified. The method is also simple in that users are required to specify only one minimum support threshold. The search avoids examining item sets containing any item whose ancestors do not have minimum support.

If the minimum support threshold is set too high, it could miss several meaningful associations occurring at low abstraction levels. If the threshold is set too low, it may generate many uninteresting associations occurring at high abstraction levels. This provides the motivation for the following approach.

Using reduced minimum support at lower levels (referred to as reduced support): Each level of abstraction has its own minimum support threshold. The lower the abstraction level is, the smaller the corresponding threshold.

1. level-by-level independent: This is a full breadth search, where no background knowledge of frequent item sets is used for pruning. Each node is examined, regardless of whether or not its parent node is found to be frequent.

2. level-cross filtering by single item: An item at the i-th level is examined if and only if its parent node at the (i - 1)-th level is frequent. If a node is frequent, its children will be examined; otherwise, its descendents are pruned from the search.

3. level-cross filtering by k-item set: A k-item set at the i-th level is examined if and only if its corresponding parent k-itemset at the (i - 1)-th level is frequent.

The level-by-level independent strategy, numerous infrequent items at low levels, finding associations. The level-cross filtering by k-itemset strategy only the children of frequent k-item sets. With this may valuable patterns may be filtered out.

**Mining multidimensional association rules from relational databases and data warehouses:**

In mining multidimensional association rules, more than one dimension or predicate is involved.

buys(X; “IBM home computer")=> buys(X; “Sony b/w printer")

where X is a variable representing customers who purchased items in All Electronics transactions. Similarly, if “printer" is a generalization of “Sony b/w printer", then a multilevel association rule like

IBM home computers => printer

buys(X; “IBM home computer") => buys(X; “printer")

can be expressed as single-dimensional or intra-dimension association rules since they each contain a single distinct predicate (e.g., buys) with multiple occurrences.

age(X, “19 … 24") ^ occupation(X, “student") =>buys(X, “laptop"):

Association rules that involve two or more dimensions or predicates can be referred to as multidimensional association rules. Above association rule contains three predicates (age, occupation, and buys), each of which occurs only once in the rule. Hence, it has no repeated predicates. Multidimensional association rules with no repeated predicates are called inter-dimension association rules. We may also be interested in mining multidimensional association rules with repeated predicates, which contain multiple occurrences of some predicate. These rules are called hybrid-dimension association rules.

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, since their values are “names of things". Quantitative attributes are numeric and have an implicit ordering among values (e.g., age, income, price).

In the first approach, quantitative attributes are discretized using predefined concept hierarchies. This discretization occurs prior to mining. For instance, a concept hierarchy for income may be used to replace the original numeric values of this attribute by ranges, such as “0-20K", “21-30K", “31-40K", and so on. Here, discretization is static and predetermined. The discretized numeric attributes, with their range values, can then be treated as categorical attributes (where each range is considered a category).

2. In the second approach, quantitative attributes are discretized into “bins" based on the distribution of the data. These bins may be further combined during the mining process. The discretization process is dynamic and established so as to satisfy some mining criteria, such as maximizing the confidence of the rules mined. Because this strategy treats the numeric attribute values as quantities rather than as predefined ranges or categories, association rules mined from this approach are also referred to as quantitative association rules.

3. In the third approach, quantitative attributes are discretized so as to capture the semantic meaning of such interval data. This dynamic discretization procedure considers the distance between data points. Hence, such quantitative association rules are also referred to as distance-based association rules.

Aquan1 ^ Aquan2 => Acat

Where Aquan1 and Aquan2 are tests on quantitative attribute ranges and Acat tests a categorical attribute from the task-relevant data. Such rules have been referred to as two-dimensional quantitative association rules, since they contain two quantitative dimensions.

age(X, “30 … 34") ^ income(X, “42K … 48K") => buys(X, “high resolution TV")

**Clustering the association rules**. The strong association rules obtained in the previous step are then mapped to a 2-D grid. 2-D quantitative association rules predicting the condition buys(X, “high resolution TV") on the rule right-hand side, given the quantitative attributes age and income.

age(X, 34) ^ income(X; “30 … 40K") => buys(X, “high resolution TV")

age(X, 35) ^ income(X; “30 … 40K") => buys(X, “high resolution TV")

age(X, 34) ^ income(X; “40 … 50K") => buys(X, “high resolution TV")

age(X, 35) ^ income(X; “40 … 50K") => buys(X, “high resolution TV")

age(X, “34 … 35") ^ income(X, “30 … 50K") => buys(X, “high resolution TV")

**SUMMARY**

The discovery of association relationships among huge amounts of data is useful in selective marketing, decision analysis, and business management. A popular area of application is market basket analysis, which studies the buying habits of customers by searching for sets of items that are frequently purchased together (or in sequence). Association rule mining consists of first finding frequent item sets (set of items, such as A and B, satisfying a minimum support threshold, or percentage of the task-relevant tuples), from which strong association rules in the form of A ) B are generated. These rules also satisfy a minimum confidence threshold

(a pre specified probability of satisfying B under the condition that A is satisfied).

\_ Association rules can be classified into several categories based on different criteria, such as:

1. Based on the types of values handled in the rule, associations can be classified into Boolean vs. quantitative.

A Boolean association shows relationships between discrete (categorical) objects. A quantitative association is a multidimensional association that involves numeric attributes which are discretized dynamically.

It may involve categorical attributes as well.

2. Based on the dimensions of data involved in the rules, associations can be classified into single-dimensional vs. multidimensional.

Single-dimensional association involves a single predicate or dimension, such as buys; whereas multidimensional association involves multiple (distinct) predicates or dimensions. Single-dimensional association shows intra-attribute relationships (i.e., associations within one attribute or dimension); whereas multidimensional association shows inter-attribute relationships (i.e., between or among attributes/dimensions).

3. Based on the levels of abstractions involved in the rule, associations can be classified into single-level vs.multi level.

In a single-level association, the items or predicates mined are not considered at different levels of abstraction, whereas a multilevel association does consider multiple levels of abstraction.

\_ The Apriori algorithm is an efficient association rule mining algorithm which explores the level-wise mining property: all the subsets of a frequent itemset must also be frequent. At the k-th iteration (for k > 1), it forms

frequent (k + 1)-itemset candidates based on the frequent k-item sets, and scans the database once to find the complete set of frequent (k + 1)-item sets, Lk+1.

Variations involving hashing and data scan reduction can be used to make the procedure more efficient. Other variations include partitioning the data (mining on each partition and them combining the results), and sampling the data (mining on a subset of the data). These variations can reduce the number of data scans required to as little as two or one.

\_ Multilevel association rules can be mined using several strategies, based on how minimum support thresholds are defined at each level of abstraction. When using reduced minimum support at lower levels, pruning approaches include level-cross-filtering by single item and level-cross filtering by k-itemset. Redundant multilevel (descendent) association rules can be eliminated from presentation to the user if their support and confidence are close to their expected values, based on their corresponding ancestor rules.

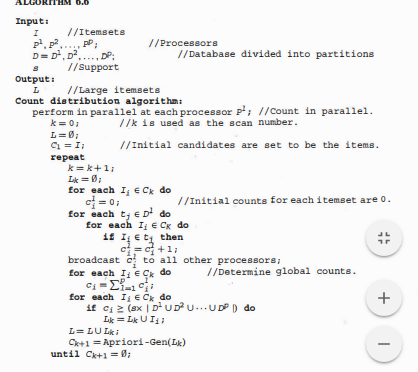
\_ Techniques for mining multidimensional association rules can be categorized according to their treatment of quantitative attributes. First, quantitative attributes may be discretized statically, based on predefined concept hierarchies. Data cubes are well-suited to this approach, since both the data cube and quantitative attributes can make use of concept hierarchies. Second, quantitative association rules can be mined where quantitative attributes are discretized dynamically based on binning, where \adjacent" association rules may be combined by clustering. Third, distance-based association rules can be mined to capture the semantics of interval data, where intervals are defined by clustering.

\_ Not all strong association rules are interesting. Correlation rules can be mined for items that are statistically correlation

**3.2 PARALLEL AND DISTRIBUTED ALGORITHMS**

Most parallel or distributed association rule algorithms strive to parallelize either the data, known as data parallelism, or the candidates, referred to iS task parallelism. With task parallelism, the candidates are partitioned and counted separately at each processor. Obviously, the partition algorithm would be easy to parallelize using the task parallelism approach. Other dimensions in differentiating different parallel association rule algorithms are the load-balancing approach used and the architecture. The data parallelism algorithms have reduced communication cost over the task, because only the initial candidates (the set of items) and the local counts at each iteration must be distributed. With task parallelism, not only the candidates but also the local set of transactions must be broadcast to all other sites. However, the data parallelism algorithms require that memory at each processor be large enough to store all candidates at each scan (otherwise the performance will degrade considerably because 1/0 is required for both the database and the candidate set). The task parallelism approaches can avoid this because only the subset of the candidates that are assigned to a processor during each scan must fit into memory. Since not all partitions of the candidates must be the same size, the task parallel algorithms can adapt to the amount of memory at each' site. The only restriction is that the total size of all candidates be small enough to fit into the total size of memory in all processors combined. Note that there are some variations of the• basic algorithms discussed in this section that address these memory issues. Performance studies have shown that the data parallelism tasks scale linearly with the number of processors and lay database size.

Data Parallelism One data parallelism algorithm is the count distribution algorithm (CDA). The data base is divided into p partitions, one for each processor. Each processor counts the candidates for its data and then broadcasts its counts to all other processors. Each processor then determines the global counts. These then are used to determine the large item sets and to generate the candidates for the next scan. The algorithm is shown in Algorithm 6.6.

****

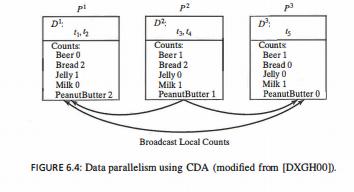
****

Figure 6.4, which is modified from [DXGHOO], illustrates the approach used by the CDA algorithm using the grocery store data. Here there are three processors. The first two transactions are counted at P 1 , the next two at P 2 , and the last one at P 3 . When the local counts are obtained, they are then broadcast to the other sites so that global counts can be generated.

Task Parallelism The data distribution algorithm (DDA) demonstrates task parallelism. Here the candidates as well as the database are partitioned among the processors. Each processor in parallel commas the candidates given to it using its local database partition. Following our convention, we use c£ to indicate the candidates of size k examined at processor P1. Also, Li are the local large k-item sets at processor l. Then each processor broadcasts its database partition to all other processors. Each processor then uses this to obtain a global count for its data and broadcasts this count to all other processors. Each processor then can determine globally large item sets and generate the next candidates. These candidates then are divided among the processors for the next scan. Algorithm 6. 7 show this approach. Here we show that the candidates are actually sent to each processor. However, some prearranged technique could be used locally by each processor to determine its own candidates. This algorithm suffers from high message traffic whose impact can be reduced by overlapping communication and processing.

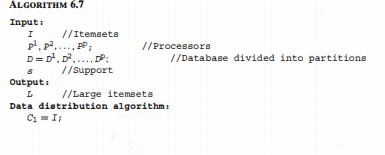
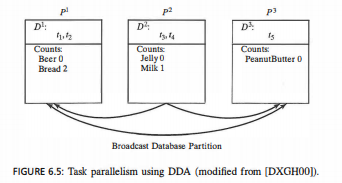
****

Figure 6.5, which is modified from [DXGHOO], illustrates the approach used by the DDA algorithm using the grocery store data. Here there are three processors. P1 is counting Beer and Bread, P2 is counting Jelly and Milk, and P 3 is counting Peanut Butter. The first two transactions initially are counted at P 1 , the next two at P 2 , and the last one at P 3 • When the local counts are obtained, the database partitions are then broadcast to the other sites so that each site can obtain a global count.

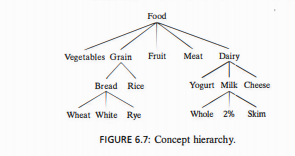
****

**4.3 INCREMENTAL RULES**

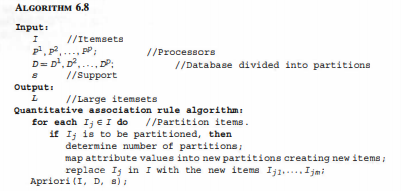
All algorithms discussed so far assume a static database. However, in reality we cannot assume this. With these prior algorithms, generating association rules for a new database state requires a complete rerun of the algorithm. Several approaches have been proposed to address the issue of how to maintain the association rules as the underlying database changes. Most of the proposed approaches have addressed the issue of how to modify the association rules as inserts are performed on the database. These incremental updating approaches concentrate on determining the large item sets for D U db where D is a database state and db are updates to it and where the large item sets for D, L is known. 1 One incremental approach, fast update (FUP), is based on the Apriori algorithm. Each iteration, k, scans both db and D with candidates generated from the prior iteration, k - 1, based on the large item sets at that scan. In addition, we use as part of the candidate set for scan k to be Lk found in D. The difference is that the number of candidates examined at each iteration is reduced through pruning of the candidates. Although other pruning techniques are used, primary pruning is based on the fact that we already know L from D. Remember that according to the large item set property, an item set must be large in at least one of these partitions of the new database. For each scan k of db, Lk plus the counts for each item set in Lk are used as input. When the count for each item in Lk is found in db, we automatically know whether it will be large in the entire database without scanning D. We need not even count any items in Lk during the scan of db if they have a subset that is not large in the entire database.

**3.4 ADVANCED ASSOCIATION RULE TECHNIQUES**

In this section we investigate several techniques that have been proposed to generate association rules that are more complex than the basic rules. 6.7.1 Generalized Association Rules using a concept hierarchy that shows the set relationship between different items, generalized association rules allow rules at different levels. Example 6. 7 illustrate the use of these generalized rules using the concept hierarchy in Figure 6.7. Association rules could be generated for any and all levels in the hierarchy. A generalized association rule, X =? Y, is defined like a regular association rule with the restriction that no item in Y may be above any item in X. When generating generalized association rules, all possible rules are generated using one or more given hierarchies. Several algorithms have been proposed to generate generalized rules. The simplest would be to expand each transaction by adding (for each item in it) all items above it in any hierarchy.

****

Multiple-Level Association Rules A variation of generalized rules are multiple-level association rules. With multiple-level rules, item sets may occur from any level in the hierarchy: Using a variation of the Apriori algorithm, the concept hierarchy is traversed in a top-down manner and large item sets are generated. When large item sets are found at level i, large item sets are generated for level i + 1. Large k- item sets at one level in the concept hierarchy are used as candidates to generate large k-item sets for children at the next level. Modification to the basic association rule ideas may be changed. We expect that there is more support for item sets occurring at higher levels in the concept hierarch!. Thus, the minimum support required for association rules may vary based on level m the hierarchy. We would expect that the frequency of item sets at higher levels i . s . much greater than the frequency of item sets at lower levels. Thus, for the reduced nurumum support concept, the following rules apply: • The minimum support for all nodes in the hierarchy at the same level is identical. • If o:; is the minimum support for level i in the hierarchy and a; -1 is the minimum support for level i - 1, then a; -1 > ct; . Quantitative Association Rules The association rule algorithms discussed so far assume that the data are categorical. A quantitative association rule is one that involves categorical and quantitative data. An example of a quantitative rule is: A customer buys wine for between $30 and $50 a bottle => she also buys caviar This differs from a traditional association rule such as: A customer buys wine => she also buys caviar. The cost quantity has been divided into an interval (much as was done when we looked at handling numeric data in clustering and classification). In these cases, the items are not simple literals. For example, instead of having the items {Bread, Jelly}, we might have the items {(Bread:[O ... 1]), (Bread:(l ... 2]) , (Bread:(2 ... oo)), (Jelly:[O . . . 1 .5]), (Jelly:(l.5 ... 3]), (Jelly:(3 ... oo))}. The basic approach to finding quantitative association rules is found in Algorithm 6.8. Here we show the J\priori algorithm being used to generate the large item sets, but any such algorithm could pe used.

****

Because we have divided what was one item into several items, the minimum support and confidence used for quantitative rules may need to be lowered. The minimum support problem obviously is worse with a large number of intervals. Thus, an alternative solution would be to combine adjacent intervals when calculating support. Similarly, when there are a small number of intervals, the confidence threshold may need to be lowered. For example, look at X => Y. Suppose there are only two intervals for X. Then the count for those transactions containing X will be quite high when compared to those containing Y (if this is a more typical item with many intervals)

Using Multiple Minimum Supports When looking at large databases with many types of data, using one minimum support value can be a problem. Different items behave differently. It certainly is easier to obtain a given support threshold with an attribute that has only two values than it is with an

attribute that has hundreds of values. It might be more meaningful to find a rule of the form Skim Milk ==> Whitbread with a support of 3% than it is to find Milk ==> Bread with a support of 6%. Thus, having only one support value for all association rules may not work well. Some useful rules could be missed. This is particularly of interest when looking at generalized association rules, but it may arise in other situations as well. Think of generating association rules from a non-market basket database. As was seen with quantitative rules, we may partition attribute values into ranges. Partitions that have a small number of values obviously will produce lower supports than those with a large number of values. If a larger support is used, we might miss out on generating meaningful association rules. This problem is called the rare item problem. If the minimum support is too high, then rules involving items that rarely occur will not be generated. If it is set too low, then too many rules may be generated, many of which (particularly for the frequently occurring items) are not important. Different approaches have been proposed to handle this. One approach is to partition the data based on support and generate association rules for each partition separately. Alternatively, we could group rare items together and generate association rules for these groupings. A more recent approach to handling this problem is to combine clustering and association rules. First we cluster the data together based on some clustering criteria, and then we generate rules for each cluster separately. This is a generalization of the partitioning of the data solution. One approach, M/S apriori, allows a different support threshold to be indicated for each item. Here MIS stands for minimum item support. The minimum support for a rule is the minimum of all the minimum supports for each item in the rule. An interesting problem occurs when multiple minimum supports are used. The minimum support requirement for an item set may be met even though it is not met for some of its subsets. This seems to violate the large item set property. Example 6.8, which is adapted from [LHM99], illustrates this. A variation of the downward closure property, called the sorted downward closure property, is satisfied and used for the MIS apriori algorithm. First the items are sorted in ascending MIS value. Then the candidate generation at scan 2 looks only at adding to a large item any item following it (larger than or equal to MIS value) in the sorted order.

Correlation Rules A correlation rule is defined as a set of item sets that are correlated. The motivation for developing these correlation rules is that negative correlations may be useful. 6.8 1 88 Chapter 6 Association Rules Example 6.9, which is modified from [BMS77], illustrates this concept. In this example, even though the probability of purchasing two items together seems high, it is much higher if each item is purchased without the other item. Correlation satisfies upward closure in the item set lattice. Thus, if a set is correlated, so is every superset of i

Example

Suppose there are two items, {A, B} where A ::::} B has a support of 15% and a confidence of 60%. Because these values are high, a typical association rule algorithm probably would deduce this to be a valuable rule. However, if the probability to purchase item B is 70%, then we see that the probability of purchasing B has actually gone down, presumably because A was purchased. Thus, there appears to be a negative correlation between buying A and buying B. The correlation can be expressed as P (A, B) correlation(A ===> B) = P (A) P (B) (6.1) 1 which in this case is: 0.2°5�50.7 = 0. 857. Because this correlation value is lower than 1, it indicates a negative correlation between A and B.

**3.5 MEASURING THE QUALITY OF RULES**

Support and confidence are the normal methods used to measure the quality of an association rule: s (A ===> B) = P (A, B) (6.2) and a (A ===> B) = P (B I A) (6.3) However, there are some problems associated with these metrics. For example, confidence totally ignores P (B). A rule may have a high support and confidence but may be an obvious rule. For example, if someone purchases potato chips, there may be a high likelihood that he or she would also buy a cola. This rule is not really of interest because it is not surprising. Various concepts such as surprise and interest have been used to evaluate the quality or usefulness of rules. We briefly examine some of these in this section. With correlation rules, we saw that correlation may be used to measure the relationship between items in a rule. This may also be expressed as the lift or interest P (A, B) interest(A ===> B) = P (A) P (B) (6.4) This measure takes into account both P (A) and P (B). A problem with this measure is that it is symmetric. Thus, there is no difference between the value for interest(A ::::} B) and the value for interest(B ::::} A). As with lift, conviction takes into account both P (A) and P (B).

Conviction measure inverts this ratio. The formula for conviction is [BMS77] . . P (A) P (-•B Conviction (A ===> B) = P(A, -. B ) (6.5) Conviction has a value of 1 if A and B are not related. Rules that always hold have a value of oo. The usefulness of discovered association rules may be tied to the amount of surprise associated with the rules or how they deviate from previously known rules. Here surprise is a measure of the changes of correlations between items over time. For example, if you are aware that beer and pretzels are often purchased together, it would be a surprise if this relationship actually lowered significantly. Thus, this rule beer ::::} pretzel would be of interest even if the confidence decreased. Another technique to measure the significance of rules by using the chi squared test for independence has been proposed. This significance test was proposed for use with correlation rules. Unlike the support or confidence measurement, the chi squared significance test takes into account both the presence and the absence of items in sets. Here it is used to measure how much an item set (potential correlation rule) count differs from the expected. The chi squared test is well understood because it has been used in the statistics community for quite a while. Unlike support and confidence, where arbitrary values must be chosen to determine which rules are of interest, the chi squared values are well understood with existing tables that show the critical values to be used to determine relationships between items. The chi squared statistic can be calculated in the following manner. Suppose the set of items is I = {!1 , h ... , I m}. Because we are interested in both the occurrence and the non-occurrence of an item, a transaction t1 can be viewed as

****

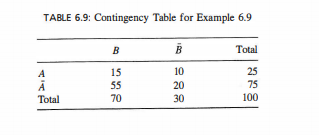
Given any possible item set X, it also is viewed as a subset of the Cartesian product. The chi squared statistic is then calculated for X as

****

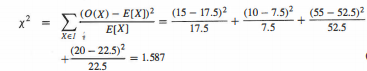
Here O (X) is the count of the number of transactions that contain the items in X. For one item Ii , the expected value is E [ Ii] = 0 Ui), the count of the number of transactions that contain Ii . E[Ji] = n - O (li). The expected value E [X] is calculated assuming independence and is thus defined as

****

Here n is the number of transactions. (6.8) Table 6.9, which is called a contingency table, shows the distribution of the data in Example 6.9 assuming that the sample has 100 items in it. From this we find

****

E[AB] = 17.5, E[AB] = 7.5, E[AB] = 52.5, and E[AB] = 22.5. Using these values, we calculate x 2 for this example as

****

If all values were independent, then the chi squared statistic should be 0. A chi squared table (found in most statistics books) can be examined to interpret this value. Examining a table of critical values for the chi squared statistic, we see that a chi squared value less than 3.84 indicates that we should not reject the independent assumption. This is done with 95% confidence. Thus, even though there appears to be a negative correlation between A and B, it is not statistically significant.