e-PGPathshala
Subject : Computer Science
Paper: Data Analytics
Module No 24: CS/DA/24 - Data Analytics – Association Rule Mining - II
Quadrant 1 – e-text

1.1 Introduction

Apriori Algorithm is one of the classic algorithm used in Data Mining to find association rules. The fundamentals of association rule mining is discussed in the previous module. This chapter will give an idea about the working of Apriori algorithm.

1.2 Learning Outcomes

- **To learn working of apriori algorithm**
- **To Understand the working using solved problems**

1.3 Mining Frequent Itemsets: the Key Step

Frequent Itemsets: The sets of item which has minimum support (denoted by L_i for i^{th}-Itemset). Frequent itemsets are a form of frequent pattern. Itemsets that meet a minimum support threshold are referred to as frequent itemsets.

A subset of a frequent itemset must also be a frequent itemset, i.e., if {AB} is a frequent itemset, both {A} and {B} should be a frequent itemset. Iteratively find frequent itemsets with cardinality from 1 to k (k-itemset) and use the frequent itemsets to generate association rules.

1.3.1 Need of frequent itemset

There are many benefits for finding frequent item sets. For example by finding frequent itemsets, a retailer can learn what is commonly bought together. Especially important are pairs or larger sets of items that occur much more frequently than would be expected were the items bought independently. We can find all combinations of items that occur together. They might be interesting ,i.e, in shop some items will place together(e.g., some combination item of food like bread and butter). We can see that frequent itemsets are only positive combinations. Frequent itemsets aims at providing a summary for the data.

However, applications of frequent-itemset analysis are not limited to market baskets. The same model can be used to mine many other kinds of data. Some examples are:
1. Related concepts: Let items be words, and let baskets be documents (e.g., Web pages, blogs, tweets). A basket/document contains those items/words that are present in the document. If we look for sets of words that appear together in many documents, the sets will be dominated by the most common words (stop words).There, even though the intent was to find snippets that talked about cats and dogs, the stop words “and” and “a” were prominent among the frequent itemsets. However, if we ignore all the most common words, then we would hope to find among the frequent pairs some pairs of words that represent a joint concept. For example, we would expect a pair like {Brad, Angelina} to appear with surprising frequency.
2. Plagiarism: Let the items be documents and the baskets be sentences. An item/document is “in” a basket/sentence if the sentence is in the document. This arrangement appears backwards, but it is exactly what we need, and we should remember that the relationship between items and baskets is an arbitrary many-many relationship. That is, “in” need not have its conventional meaning: “part of.” In this application, we look for pairs of items that appear together in several baskets. If we find such a pair, then we have two documents that share several sentences in common. In practice, even one or two sentences in common is a good indicator of plagiarism.
3. Biomarkers: Let the items be of two types – biomarkers such as genes or blood proteins, and diseases. Each basket is the set of data about a patient: their genome and blood-chemistry analysis, as well as their medical history of disease. A frequent itemset that consists of one disease and one or more biomarkers suggests a test for the disease.

1.3.2 Finding frequent sets

- Task: Given a transaction database D and a min_sup threshold find all frequent itemsets and the frequency of each set in this collection
- Stated differently: Count the number of times combinations of attributes occur in the data. If the count of a combination is above min_sup report it.
- Recall: The input is a transaction database D where every transaction consists of a subset of items from some universe

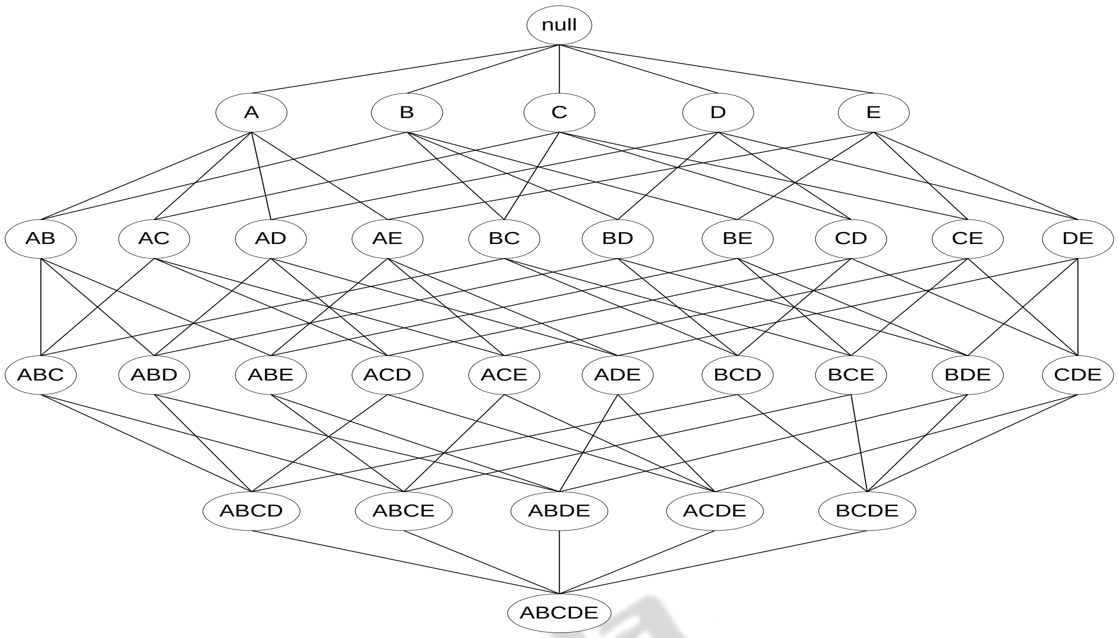


Figure 1. Generating k-itemsets

Given d items, then there are 2^d possible itemsets which can be generated as shown in figure 1. If **min_sup = 0**, then all subsets of ***I*** will be frequent and thus the size of the collection will be very large. This summary is very large (maybe larger than the original input) and thus not interesting. The task of finding all frequent sets is interesting typically only for relatively large values of **min_sup.** In the above figure 1if item B is not frequent, then all the supersets involving item B will also be infrequent as shown in figure 2. This principle is called apriori principle. Hence such non-frequent itemsets may be pre or post pruned.

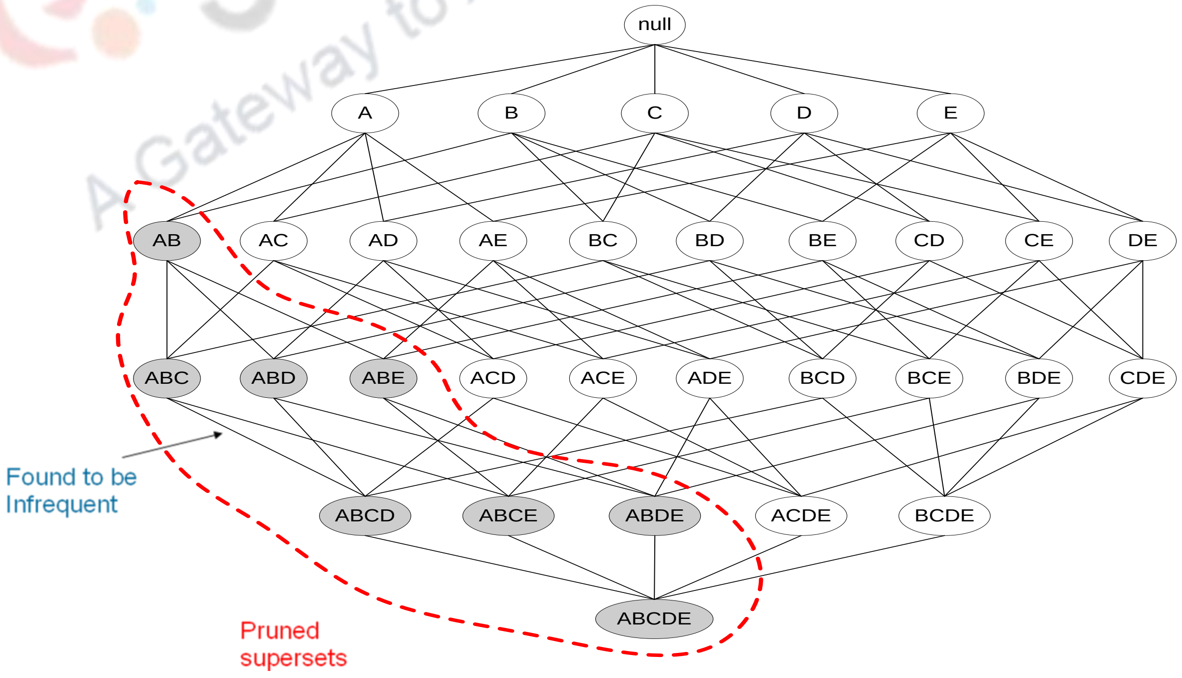


Figure 2. Pruning inf requent itemsets

1.4 Apriori Algorithm

Together with the introduction of the frequent set mining problem, also the first algorithm to solve it was proposed, later denoted as AIS. Shortly after that the algorithm was improved by R. Agrawal and R. Srikant and called Apriori. It is a seminal algorithm, which uses an iterative approach known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets.

It uses the Apriori property to reduce the search space: All nonempty subsets of a frequent itemset must also be frequent.

P(I)<min_sup => I is not frequent

P(I+A)<min_sup => I+A is not frequent either

Antimonotone property – if a set cannot pass a test, all of its supersets will fail the same test as well

In the next section we will see how the apriori property is used in the Apriori algorithm: Let us look at how L_{k-1} is used to find L_k, for k>=2. We can distinct two steps: join and prune.

1. Join
   - finding L_k, a set of candidate k-itemsets is generated by joining L_{k-1} with itself
   - The items within a transaction or itemset are sorted in lexicographic order
   - For the (k-1) itemset: $l_i[1]<l_i[2]<\ldots<l_i[k-1]$
   - The members of Lk-1 are joinable if their first(k-2) items are in common
   - Members l_1, l_2 of L_{k-1} are joined if $(l_1[1]=l_2[1])$ and $(l_1[2]=l_2[2])$ and.... and $(l_1[k-2]=l_2[k-2])$ and $(l_1[k-1]<l_2[k-1])$ – no duplicates
   - The resulting itemset formed by joining l_1 and l_2 is $l_1[1]$, l1[2],…, $l_1[k-2]$, $l_1[k-1]$, $l_2[k-1]$
2. Prune
   - C_k is a superset of L_k, L_k contain those candidates from C_k, which are frequent
   - Scanning the database to determine the count of each candidate in C_k – heavy computation
   - To reduce the size of C_k the Apriori property is used: if any (k-1) subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either,so it can be removed from C_k. – subset testing (hash tree)

Let us take the following 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

The join and prune steps for this example:

Scan D for count of each candidate

C_1: I1 – 6, I2 – 7, I3 -6, I4 – 2, I5 - 2

Compare candidate support count with minimum support count (min_sup=2)

L_1: I1 – 6, I2 – 7, I3 -6, I4 – 2, I5 - 2

Generate C2 candidates from L1 and scan D for count of each candidate

C_2: {I1,I2} – 4, {I1, I3} – 4, {I1, I4} – 1, …

Compare candidate support count with minimum support count

L_2: {I1,I2} – 4, {I1, I3} – 4, {I1, I5} – 2, {I2, I3} – 4, {I2, I4} - 2, {I2, I5}– 2

Generate C_3 candidates from L_2 using the join and prune steps:

Join: C_3=L_2 x L_2={{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2,I3, I5}, {I2, I4, I5}}

Prune: C_3: {I1, I2, I3}, {I1, I2, I5}

Scan D for count of each candidate

C_3: {I1, I2, I3} - 2, {I1, I2, I5} – 2

Compare candidate support count with minimum support count

L_3: {I1, I2, I3} – 2, {I1, I2, I5} – 2

Generate C_4 candidates from L_3

C_4=L_3 x L_3={I1, I2, I3, I5}

This itemset is pruned, because its subset {{I2, I3, I5}} is not frequent =>C_4=null

1.4.2 Challenges in Counting Supports of Candidates

In reality, the total number of candidates can be very huge or sometimes one transaction may contain many candidates. In such scenarios, the candidate itemsets can be are stored in a hash-tree. The leaf node of hash-tree contains a list of itemsets and counts, the interior node contains a hash table. A subset function may be defined to find all the candidates contained in a transaction.

1.5 Association rule

Association rules problem is as follows: Let I = {i1,i2,...in} be a set of literals call items. Let D be a set of all transactions where each transaction T is a set of items such that $T \subseteq I$. Let X, Y be a set of items such that $X, Y \subseteq I$. An association rule is an implication in the form $X \Rightarrow Y$, where $X \subset I, Y \subset I$, $X \cap Y = \emptyset$.

1.5.1 Measures of Association Rules

Association rules are created by analyzing data for frequent if - then patterns and using the criteria support and confidence to identify the most important relationships. Support (s) is an indication of how frequently the items appear in the database. Confidence (c) indicates the number of times the if/then statements have been found to be true.

If the association is denoted by $A \rightarrow B$ [s, c]

Support (s) :- Fraction of transactions that contain both A and B, i.e., it denotes the frequency of the rule within transactions. A high value means that the rule involves a great part of database

$$\text{support } (A \rightarrow B \text{ [s, c]}) = p(A \cup B)$$

Confidence (c): - Measures how often items in Y appear in transactions that contain X.

T denotes set if Transactions transactions or it denotes the percentage of transactions containing A which also contain B. It is an estimation of conditioned probability .

$$\text{confidence}(A \rightarrow B \text{ [s, c]}) = p(B|A) = \text{sup}(A,B)/\text{sup}(A).$$

1.5.2 Compact Representation of Frequent Itemset

When you have a large market basket data with over a hundred items, the number of frequent itemsets grows exponentially and this in turn creates an issue with storage. And it is for this purpose that alternative representations have been derived which reduce the initial set but can be used to generate all other frequent itemsets. The

Maximal and Closed Frequent Itemsets are two such representations that are subsets of the larger frequent itemset that will be discussed.

Maximal Frequent Itemset is a frequent itemset for which none of its immediate supersets are frequent.

Steps for identifiying Maximal Frequent Itemset

1. Examine the frequent itemsets that appear at the border between the infrequent and frequent itemsets.
2. Identify all of its immediate supersets.
3. If none of the immediate supersets are frequent, the itemset is maximal frequent.

For instance consider the diagram shown in figure 3, the lattice is divided into two groups, red dashed line serves as the dermarcation, the itemsets above the line that are blank are frequent itemsets and the blue ones below the red dashed line are infrequent.

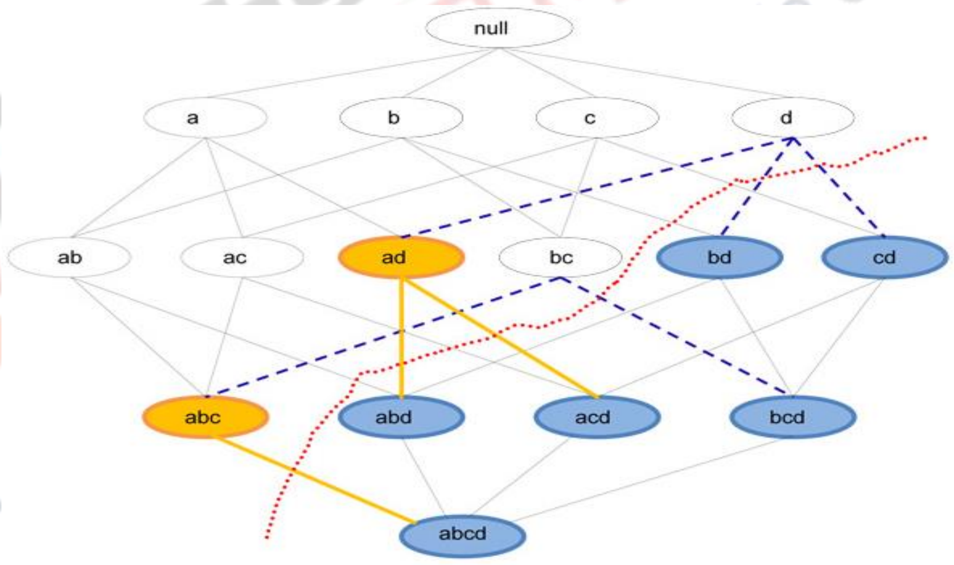


Figure 3. Lattice highlighting maximal frequent itemset

- In order to find the maximal frequent itemset, you first identify the frequent itemsets at the border namely *d, bc, ad* and *abc*.
- Then identify their immediate supersets,
  the supersets for *d, bc* are characterized by the blue dashed line and if you trace the lattice you notice that for d, there are three supersets and one of them, ad is frequent and this can't be maximal frequent,
  for *bc* there are two supersets namely *abc* and *bcd abc* is frequent and so *bc* is NOT maximal frequent.

- The supersets for ad and abc are characterized by a solid orange line, the superset for *abc* is *abcd* and being that it is infrequent, *abcd* is maximal frequent. For ad, there are two supersets *abd* and *acd*, both of them are infrequent and so ***ad*** is also maximal frequent.

Closed Frequent Itemset

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

1. First identify all frequent itemsets.
2. Then from this group find those that are closed by checking to see if there exists a superset that has the same support as the frequent itemset, if there is, the itemset is disqualified, but if none can be found, the itemset is closed. *An alternative method is to first identify the closed itemsets and then use the minsup to determine which ones are frequent.*

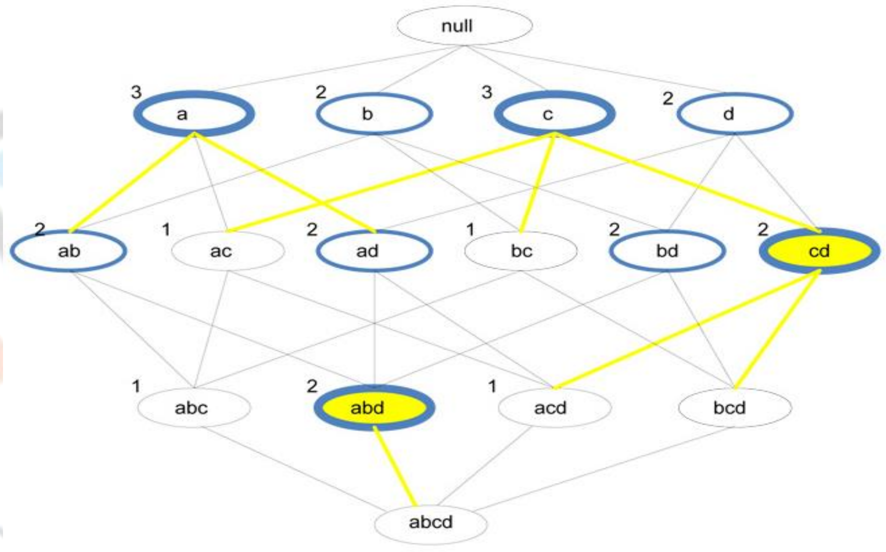


Figure 4. Lattice highlighting closed frequent itemset

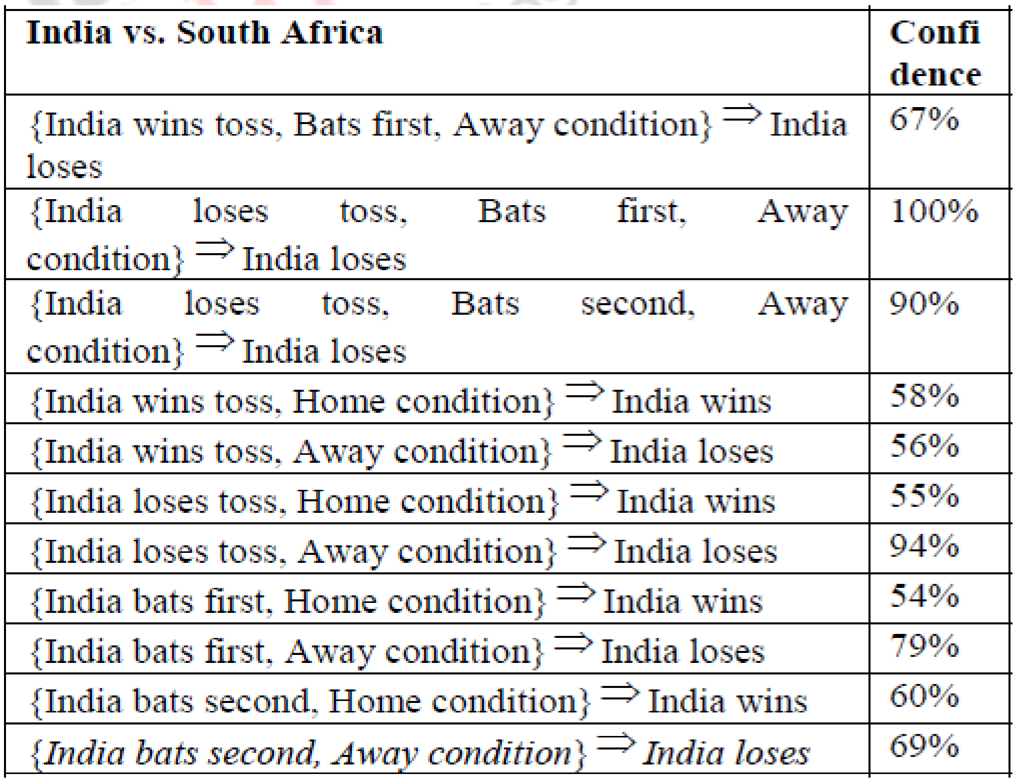
The lattice diagram in figure 4 highlights the maximal, closed and frequent itemsets. The itemsets that are circled with blue are the frequent itemsets. The itemsets that are circled with the thick blue are the closed frequent itemsets. The itemsets that are circled with the thick blue and have the yellow fill are the maximal frequent itemsets. In order to determine which of the frequent itemsets are closed, all you have to do is check to see if they have the same support as their supersets, if they do they are not closed.

For example **ad** is a frequent itemset but has the same support as **abd** so it is NOT a closed frequent itemset; c on the other hand is a closed frequent itemset because all of its supersets, ac, bc, and cd have supports that are less than 3. As you can see there are a total of 9 frequent itemsets, 4 of them are closed frequent itemsets and out of these 4, 2 of them are maximal frequent itemsets. This brings us to the relationship between the three representations of frequent itemsets.

Case Studies

A) "Application of Association Rule Mining: A Case Study on Team India", International Conference on Computer Communication and Informatics, 2013.

Association Rule Mining algorithm is applied to sports management, especially mining relationship from data on performance of Indian cricket team in one day international (ODI) matches. This analysis helped in determining factors associated with the match outcome so as to enable the team to formulate match winning strategies. Data for this analysis was obtained from the cricinfo website on the matches played by India against all the other test playing nations from 1974 till 2013 and data from other secondary sources were also considered to obtain deeper insights on playing conditions and the match outcome.



India vs. South Africa	Confidence
{India wins toss, Bats first, Away condition} $\Rightarrow$ India loses	67%
{India loses toss, Bats first, Away condition} $\Rightarrow$ India loses	100%
{India loses toss, Bats second, Away condition} $\Rightarrow$ India loses	90%
{India wins toss, Home condition} $\Rightarrow$ India wins	58%
{India wins toss, Away condition} $\Rightarrow$ India loses	56%
{India loses toss, Home condition} $\Rightarrow$ India wins	55%
{India loses toss, Away condition} $\Rightarrow$ India loses	94%
{India bats first, Home condition} $\Rightarrow$ India wins	54%
{India bats first, Away condition} $\Rightarrow$ India loses	79%
{India bats second, Home condition} $\Rightarrow$ India wins	60%
{India bats second, Away condition} $\Rightarrow$ *India loses*	69%

The association among factors such as outcome of toss, playing in a home ground or playing abroad, batting first or batting second, and the match outcome, i.e., win or loss is examined. The outcome of the analysis revealed that Team India has performed well in the last ten years (since 2001 to 2010) as compared to entire period since the team started playing its first match (since 1974 to 2010). The mined sample association rules are as given above with the corresponding confidence values.

Summary

- In real world associations between item pairs provide great insight to business
- Major challenge in apriori is dealing with compacting the output of itemsets