Frequent pattern mining

Frequent pattern mining in data mining is the process of identifying patterns or associations within a dataset that occur frequently. This is typically done by analyzing large datasets to find items or sets of items that appear together frequently.

different algorithms used for frequent pattern mining,

1. Apriori algorithm: This is one of the most commonly used algorithms for frequent pattern mining. It uses a “bottom-up” approach to identify frequent itemsets and then generates association rules from those itemsets.
2. ECLAT algorithm: This algorithm uses a “depth-first search” approach to identify frequent itemsets. It is particularly efficient for datasets with a large number of items.
3. FP-growth algorithm: This algorithm uses a “compression” technique to find frequent patterns efficiently. It is particularly efficient for datasets with a large number of transactions.
4. Frequent pattern mining has many applications, such as Market Basket Analysis, Recommender Systems, Fraud Detection, and many more.

Advantages:

1. It can find useful information which is not visible in simple data browsing

2. It can find interesting association and correlation among data items

Disadvantages:

1. It can generate a large number of patterns
2. With high dimensionality, the number of patterns can be very large, making it difficult to interpret the results.

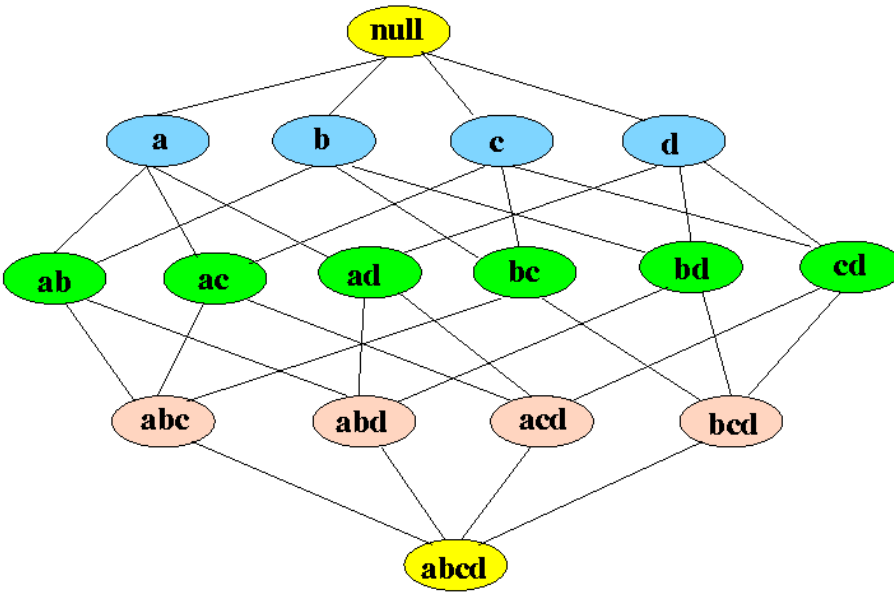
Frequent item set
A frequent item set is a set of items that occur together frequently in a dataset. The frequency of an item set is measured by the support count, which is the number of transactions or records in the dataset that contain the item set. For example, if a dataset contains 100 transactions and the item set {milk, bread} appears in 20 of those transactions, the support count for {milk, bread} is 20.

the *min sup* threshold, a hyper-parameter with high importance, which has to be set carefully by the user according to their expectations of theresults:

- Setting it to a very low value would give a large number of itemsets that would be too specific to be considered "frequent". These itemsets might apply in too few cases to be useful.

- On the other hand, very high values for *min sup* would give a small number of itemsets. These would be too generic to be useful. Thus, the resulting information would probably not represent new knowledge for the user.

Another important aspect of the *min_sup* value is whether the number of frequent itemsets that results is small enough for subsequent analysis.



Apriori

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

Algorithm Apriori.

1: INPUT the transactional dataset
2: INPUT min_sup the minimum support threshod

3: Set k = 1

4: Set $stop$ = $false$

5: **repeat**

6: Select all frequent itemsets of length k (with support at least min_sup)
7: **if** there are no two frequent itemsets of length k **then**
8: $stop = true$
9: **else**
10: Set $k = k + 1$
11: **until** stop

Components of Apriori algorithm

The given three components comprise the apriori algorithm.

1. Support
2. Confidence
3. Lift

Support

Support refers to the default popularity of any product. You find the support as a quotient of the division of the number of transactions comprising that product by the total number of transactions. Hence, we get

Support (Biscuits) = (Transactions relating biscuits) / (Total transactions)

= 400/4000 = 10 percent.

Confidence

Confidence refers to the possibility that the customers bought both biscuits and chocolates together. So, you need to divide the

number of transactions that comprise both biscuits and chocolates by the total number of transactions to get the confidence.

Hence,

Confidence = (Transactions relating both biscuits and Chocolate) / (Total transactions involving Biscuits)

= 200/400

= 50 percent.

= 400/4000 = 10 percent.

Lift

Consider the above example; lift refers to the increase in the ratio of the sale of chocolates when you sell biscuits. The mathematical equations of lift are given below.

Lift = (Confidence (Biscuits - chocolates)/ (Support (Biscuits)

= 50/10 = 5

The Apriori Algorithm makes the given assumptions

- All subsets of a frequent itemset must be frequent.
- The subsets of an infrequent item set must be infrequent.
- Fix a threshold support level. In our case, we have fixed it at 50 percent.

Advantages of Apriori Algorithm

- It is used to calculate large itemsets.

- Simple to understand and apply.

Disadvantages of Apriori Algorithms

- Apriori algorithm is an expensive method to find support since the calculation has to pass through the whole database.
- Sometimes, you need a huge number of candidate rules, so it becomes computationally more expensive.

ECLAT algorithm

The ECLAT algorithm stands for **Equivalence Class Clustering and bottom-up Lattice Traversal**. It is one of the popular methods of Association Rule mining. It is a more efficient and scalable version of the Apriori algorithm. While the Apriori algorithm works in a horizontal sense imitating the Breadth-First Search of a graph, the ECLAT algorithm works in a vertical manner just like the Depth-First Search of a graph.

Advantages over Apriori algorithm:-

1. **Memory Requirements:** Since the ECLAT algorithm uses a Depth-First Search approach, it uses less memory than Apriori algorithm.
2. **Speed:** The ECLAT algorithm is typically faster than the Apriori algorithm.
3. **Number of Computations:** The ECLAT algorithm does not involve the repeated scanning of the data to compute the individual support values.

FP Growth Algorithm?

The FP-Growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance. For so much, it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

This algorithm works as follows:

- First, it compresses the input database creating an FP-tree instance to represent frequent items.
- After this first step, it divides the compressed database into a set of conditional databases, each associated with one frequent pattern.
- Finally, each such database is mined separately.